ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
“МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРИБОРОСТРОЕНИЯ И ИНФОРМАТИКИ”

Кафедра “Персональные компьютеры и сети”

Б.М. Михайлов, Р.Ф. Халабия

КЛАССИФИКАЦИЯ И ОРГАНИЗАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Учебное пособие

Москва 2010

УДК 681.3.06
ББК 32.973

Рекомендовано в качестве методических указаний редакционно-издательским советом МГУПИ

Рецензенты:

к.т.н., доцент Брейман А.Д.

Михайлов Б.М., Халабия Р.Ф. Классификация и организация вычислительных систем: Учебное пособие. – М.: МГУПИ, 2010. - 144 с.

Пособие предназначено для студентов, обучающихся по направлению «Информатика и вычислительная техника» по специальности «Вычислительные машины, комплексы, системы и сети» по дисциплинам «Организация ЭВМ и систем» и «Персональные компьютеры и сети».

Представлены вопросы классификации и организации вычислительных систем. Основное внимание уделено архитектурам SIMD и MIMD, а также рассмотрен вопрос перспективных направлений развития вычислительных систем.

.

Содержание

Введение

Без вычислительных машин, или компьютеров, в настоящее время невозможна ни одна сфера человеческой деятельности. Компьютеры стали частью не только сферы производства, но и домашнего быта. Множество людей проводят часы за экраном своего компьютера, получая последние новости, биржевые сводки, цены, технические сведения, прогнозы погоды и многое другое из сети Интернет, а также используют компьютер для игр и развлечений.

Термин «электронная вычислительная машина», или ЭВМ, совершенно не означает, что она предназначена только для каких-либо вычислений. Это уже не просто вычислительные машины, а системы обработки данных, способные хранить информацию, редактировать, обновлять, выполнять сортировку и поиск нужных данных, формировать таблицы, диаграммы и отчеты, осуществлять логические преобразования, выдачу результатов и т. п. По этой причине в настоящее время вычислительную машину (особенно персональную) принято называть английским термином «компьютер».

Повсеместное внедрение компьютеров в современную жизнь, регулярное обновление их аппаратных и программных средств, постоянная модернизация и появление все новых компонентов требуют глубокого знания принципов их работы. Многие из нас, покупая «готовый» компьютер, не пытаются узнать, как он устроен. Больше всего покупателя интересуют стандартные программные средства и их возможности для удовлетворения своих потребностей. Но невозможно понять работу программных средств, не обладая хотя бы минимальными знаниями об аппаратуре.

Небольшие по габаритным размерам, но обладающие достаточной вычислительной мощностью компьютеры стали широко доступными. Объединение нескольких таких машин между собой, т. е. создание компьютерных сетей, обеспечило получение самой разнообразной информации. В настоящее время компьютерами пользуется все цивилизованное

человечество, это стало возможным благодаря дружественым программам и интерфейсам, ставшим непременной частью компьютера. Получила развитие всемирная сеть Интернет, объединившая средства вычислительной техники и техники связи и дающая возможность узнавать и сообщать последние новости. Теперь практически все машины тем или иным способом взаимодействуют с глобальными и локальными сетями, т. е. компьютеры становятся компонентами все более крупных систем.

Для подключения к такой сети не требуется сложных и дорогостоящих аппаратных средств — достаточно простых и недорогих адаптеров, модемов и каналов передачи данных. При этом скорость получения информации будет определяться быстродействием самой машины и пропускной способностью каналов связи, работой программ, размещением и надежностью всех компонентов, участвующих в доставке информации пользователю.

1 Общая классификация вычислительных систем

1.1 Признаки классификации вычислительных систем

Существует большое количество признаков, по которым классифицируют вычислительные системы [1,2]: по целевому назначению и выполняемым функциям, типам и числу ЭВМ или процессоров, архитектуре системы, режимам работы, методам управления элементами системы, степени разобщенности элементов вычислительной системы и др.

По **назначению** вычислительные системы делят на универсальные и специализированные. *Универсальные* ВС, или ВС *общего назначения,* предназначены для решения широкого круга задач, состав которого заранее не определен. *Специализированные ВС* ориентированы на решение узкого, заранее определенного класса задач. Специализацию ВС могут обеспечивать различные средства:

• сама структура системы (количество и типы параллельно работающих функциональных элементов, связи между ними и т.д.) может учитывать особенности выполняемых операций: матричные вычисления (процессор, в составе команд которого есть команды с векторными операциями, называют *векторным),* решение алгебраических, дифференциальных и интегральных уравнений, операции быстрого преобразования Фурье (такой процессор относят к процессорам цифровой обработки сигналов) и т.п. Практика разработки таких ВС показала, что чем уже класс задач, для решения которых предназначается специализированная ВС, тем большую производительность можно обеспечить при сокращении затрат ресурсов;

• специализация ВС может определяться наличием в составе ее периферийных устройств специального оборудования, для работы с которым требуется специальное программное обеспечение.

По виду составляющих элементов вычислительные системы принято подразделять на многомашинные и многопроцессорные ВС. *Многомашинная* ВС (ММВС) содержит несколько ЭВМ, каждая из которых имеет свою оперативную память и работает под управлением своей операционной системы. Обмен между ЭВМ идет с помощью специальных дополнительных программных и аппаратурных

средств. Работа в условиях параллелизма множества объектов — наиболее целесообразная область применения ММВС.

Если в качестве элементов ВС используются процессоры, то такая ВС относится к классу многопроцессорных. *Многопроцессорная* ВС (МПВС) — эффективное средство решения задач с параллелизмом входных данных; ее также можно использовать и при параллелизме множества задач.

В зависимости от **типов ЭВМ** или **процессоров,** из которых состоит ВС, различают однородные и неоднородные системы. В составе *однородных* систем — однотипные ЭВМ или процессоры, в составе *неоднородных* — разнотипные. В однородных системах значительно упрощается разработка и обслуживание технических и программных (в основном ОС) средств. В них обеспечивается возможность стандартизации и унификации соединений и процедур взаимодействия элементов системы, упрощается обслуживание систем, облегчается модернизация и их развитие. Вместе с тем существуют и неоднородные ВС, в которых комплексируемые элементы очень сильно отличаются по своим техническим и функциональным характеристикам и могут представлять собой специализированные процессоры.

По **характеру пространственного распределения** элементов ВС делятся на системы *сосредоточенного (локального)* и *распределенного* типов. Обычно такое деление касается только ММВС: в этом классе можно найти вычислительные системы как распределенного, так и локального типов. Как правило, многопроцессорные системы относятся к системам локального типа. Более того, благодаря успехам микроэлектроники локализация ресурсов в микропроцессорах может быть очень глубокой: в перспективных СБИС появляется возможность иметь на одном кристалле несколько параллельно работающих процессоров.

Если взаимодействие ЭВМ в составе многомашинной ВС распределенного типа организуется с помощью специальных линий связи, то такую ВС называют *вычислительной сетью.*

В локальных и распределенных ММВС сильно различается оперативность взаимодействия в зависимости от удаленности ЭВМ: время передачи информации между соседними ЭВМ, соединенными простым кабелем, может быть много меньше времени передачи данных по каналам связи.

По **методам управления элементами** различают ВС централизованные, децентрализованные и со смешанным управлением. Каждая из этих структур имеет определенные достоинства и недостатки по отношению к структурам других типов.

В *централизованных* ВС управление системой возлагается на одну главную — диспетчерскую — ЭВМ или процессор. Ее задачей является распределение нагрузки между элементами, выделение ресурсов, контроль состояния ресурсов, координация взаимодействия. Централизованные системы имеют более простые ОС. Однако выход из строя управляющей машины-диспетчера полностью парализует работу всей системы; кроме того, в централизованной ВС усложняется процесс отладки. Это ее недостатки.

В *децентрализованных системах* функции управления распределены между ее элементами. Каждая ЭВМ или процессор системы сохраняет известную автономию, а необходимое взаимодействие между элементами устанавливается по специальным наборам сигналов. С развитием ММВС, и в частности сетей ЭВМ, интерес к децентрализованным системам постоянно растет.

В системах со *смешанным управлением* совмещаются процедуры централизованного и децентрализованного управлений. Перераспределение функций осуществляется в ходе вычислительного процесса исходя из сложившейся ситуации.

По **принципу закрепления вычислительных функций** за отдельными ЭВМ или процессорами различают системы с *жестким* и *плавающим закреплением функций.* Плавающее закрепление функций обеспечивает высокую гибкость и надежность функционирования системы, но это связано с дополнительными затратами ресурсов на решение задачи динамического размещения программных модулей и массивов данных.

По режиму работы различают вычислительные системы, работающие в *оперативном* и *неоперативном временных режимах.* Оперативный режим ВС характеризуется тем, что время реализации алгоритма представляет собой фактор, определяющий эффективность системы. В неоперативных ВС такой зависимости нет.

В свою очередь в оперативных ВС выделяют системы, работающие в реальном времени — «мягком» и «жестком». Понятие реального времени подчеркивает, что допустимое время реализации алгоритма определяется динамикой объекта управления. В системах «мягкого» реального времени допускаются (но достаточно редко) ситуации, когда временные ограничения на время реализации алгоритма не выполняются. Для систем «жесткого» реального времени справедливо утверждение: «Правильное решение, полученное с опозданием, есть ошибочное решение» и это правило работает всегда, без каких бы то ни было исключений.

Вычислительные системы реального времени характеризуются, как правило, повышенными требованиями к надежности функционирования и высокой степенью автоматизации процедур подготовки входных данных.

1.2 Основные характеристики вычислительных систем

Появление любого нового направления в вычислительной технике определяется требованиями компьютерного рынка. Поэтому у разработчиков компьютеров нет одной единственной цели, одна из которых повышение основных характеристик:

- отношение стоимость/производительность;
- надежность и отказоустойчивость;
- масштабируемость;
- совместимость и мобильность программного обеспечения.

Большая универсальная вычислительная машина (мейнфрейм) или суперкомпьютер стоят дорого. Для достижения поставленных целей при проектировании высокопроизводительных конструкций приходится игнорировать стоимостные характеристики. Суперкомпьютеры фирмы Cray Research и высокопроизводительные мейнфреймы компании IBM относятся именно к этой категории компьютеров. Другим крайним примером может служить низкостоимостная конструкция, где производительность принесена в жертву для достижения низкой стоимости. К этому направлению относятся персональные компьютеры различных клонов IBM PC. Между этими двумя крайними направлениями находятся конструкции, основанные на отношении стоимость/ производительность, в которых разработчики находят баланс между стоимостными

параметрами и производительностью. Типичными примерами такого рода компьютеров являются миникомпьютеры и рабочие станции.

Для сравнения различных компьютеров между собой обычно используются стандартные методики измерения производительности. Эти методики позволяют разработчикам и пользователям использовать полученные в результате испытаний количественные показатели для оценки тех или иных технических решений, и в конце концов именно производительность и стоимость дают пользователю рациональную основу для решения вопроса, какой компьютер выбрать.

Важнейшей характеристикой вычислительных систем является *надежность*. Повышение надежности основано на принципе предотвращения неисправностей путем снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью интеграции, снижения уровня помех, облегченных режимов работы схем, обеспечение тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратуры.

Отказоустойчивость - это такое свойство вычислительной системы, которое обеспечивает ей, как логической машине, возможность продолжения действий, заданных программой, после возникновения неисправностей. Введение отказоустойчивости требует избыточного аппаратного и программного обеспечения. Направления, связанные с предотвращением неисправностей и с отказоустойчивостью, - основные в проблеме надежности. Концепции параллельности и отказоустойчивости вычислительных систем естественным образом связаны между собой, поскольку в обоих случаях требуются дополнительные функциональные компоненты. Поэтому, собственно, на параллельных вычислительных системах достигается как наиболее высокая производительность, так и, во многих случаях, очень высокая надежность. Имеющиеся ресурсы избыточности в параллельных системах могут гибко использоваться как для повышения производительности, так и для повышения надежности. Структура многопроцессорных и многомашинных систем приспособлена к автоматической реконфигурации и обеспечивает возможность продолжения работы системы после возникновения неисправностей.

Следует помнить, что понятие надежности включает не только аппаратные средства, но и программное обеспечение. Главной целью повышения надежности систем является целостность хранимых в них данных.

Масштабируемость представляет собой возможность наращивания числа и мощности процессоров, объемов оперативной и внешней памяти и других ресурсов вычислительной системы. Масштабируемость должна обеспечиваться архитектурой и конструкцией компьютера, а также соответствующими средствами программного обеспечения.

Добавление каждого нового процессора в действительно масштабируемой системе должно давать прогнозируемое увеличение производительности и пропускной способности при приемлемых затратах. Одной из основных задач при построении масштабируемых систем является минимизация стоимости расширения компьютера и упрощение планирования. В идеале добавление процессоров к системе должно приводить к линейному росту ее производительности. Однако это не всегда так. Потери производительности могут возникать, например, при недостаточной пропускной способности шин из-за возрастания трафика между процессорами и основной памятью, а также между памятью и устройствами ввода/вывода. В действительности реальное увеличение производительности трудно оценить заранее, поскольку оно в значительной степени зависит от динамики поведения прикладных задач.

Возможность масштабирования системы определяется не только архитектурой аппаратных средств, но зависит от заложенных свойств программного обеспечения. Масштабируемость программного обеспечения затрагивает все его уровни от простых механизмов передачи сообщений до работы с такими сложными объектами как мониторы транзакций и вся среда прикладной системы. В частности, программное обеспечение должно минимизировать трафик межпроцессорного обмена, который может препятствовать линейному росту производительности системы. Аппаратные средства (процессоры, шины и устройства ввода/вывода) являются только частью масштабируемой архитектуры, на которой программное обеспечение может обеспечить предсказуемый рост производительности. Важно понимать, что простой переход, например, на более мощный процессор может

привести к перегрузке других компонентов системы. Это означает, что действительно масштабируемая система должна быть сбалансирована по всем параметрам.

Концепция программной совместимости впервые в широких масштабах была применена разработчиками системы IBM/360. Основная задача при проектировании всего ряда моделей этой системы заключалась в создании такой архитектуры, которая была бы одинаковой с точки зрения пользователя для всех моделей системы независимо от цены и производительности каждой из них. Огромные преимущества такого подхода, позволяющего сохранять существующий задел программного обеспечения при переходе на новые (как правило, более производительные) модели были быстро оценены как производителями компьютеров, так и пользователями и начиная с этого времени практически все фирмы-поставщики компьютерного оборудования взяли на вооружение эти принципы, поставляя серии совместимых компьютеров. Следует заметить, что со временем даже самая передовая архитектура неизбежно устаревает и возникает потребность внесения радикальных изменений архитектуру и способы организации вычислительных систем.

В настоящее время одним из наиболее важных факторов, определяющих современные тенденции в развитии информационных технологий, является ориентация компаний-поставщиков компьютерного оборудования на рынок прикладных программных средств. Это объясняется прежде всего тем, что для конечного пользователя в конце концов важно программное обеспечение, позволяющее решить его задачи, а не выбор той или иной аппаратной платформы. Переход от однородных сетей программно совместимых компьютеров к построению неоднородных сетей, включающих компьютеры разных фирм-производителей, в корне изменил и точку зрения на саму сеть: из сравнительно простого средства обмена информацией она превратилась в средство интеграции отдельных ресурсов - мощную распределенную вычислительную систему, каждый элемент которой (сервер или рабочая станция) лучше всего соответствует требованиям конкретной прикладной задачи.

Этот переход выдвинул ряд новых требований. Прежде всего такая вычислительная среда должна позволять гибко менять количество и состав аппаратных средств и программного обеспечения в соответствии с меняющимися требованиями решаемых задач. Во-вторых, она должна обеспечивать возможность

запуска одних и тех же программных систем на различных аппаратных платформах, т.е. обеспечивать мобильность программного обеспечения. В третьих, эта среда должна гарантировать возможность применения одних и тех же человеко-машинных интерфейсов на всех компьютерах, входящих в неоднородную сеть. В условиях жесткой конкуренции производителей аппаратных платформ и программного обеспечения сформировалась концепция открытых систем, представляющая собой совокупность стандартов на различные компоненты вычислительной среды, предназначенных для обеспечения мобильности программных средств в рамках неоднородной, распределенной вычислительной системы.

Одним из вариантов моделей открытой среды является модель OSE (Open System Environment), предложенная комитетом IEEE POSIX. На основе этой модели национальный институт стандартов и технологии США выпустил документ "Application Portability Profile (APP). The U.S. Government's Open System Environment Profile OSE/1 Version 2.0", который определяет рекомендуемые для федеральных учреждений США спецификации в области информационных технологий, обеспечивающие мобильность системного и прикладного программного обеспечения. Все ведущие производители компьютеров и программного обеспечения в США в настоящее время придерживаются требований этого документа.

1.3 Режимы работы вычислительных систем

Вычислительные системы могут функционировать в следующих режимах:

- однопрограммный;
- мультипрограммный;
- пакетной обработки;
- разделения во времени;
- диалоговый;
- режим реального времени.

Однопрограммный режим работы - это режим, при котором выполняется не более одной независимой программы.

При таком режиме работы ЭВМ решение задачи начинается с загрузки программы в ОП, после чего ЭВМ последовательно выполняет команды программы.

При этом в каждый момент времени работает одно ее устройство, в то время как остальные простаивают в ожидании окончания ранее начатого действия. Значительные потери рабочего времени ЭВМ связаны с медленной работой устройства ввода-вывода по сравнению с работой быстродействующих устройств (АЛУ, ЦУУ, ОЗУ и т.д.).

Мультипрограммный режим работы - это режим, при котором в памяти ЭВМ хранится несколько программ и выполнение одной программы может быть прервано для перехода к выполнению другой с последующим возвратом к прерванной программе.

При совместном выполнении нескольких программ простои оборудования уменьшаются, поскольку увеличивается вероятность того, что среди находящихся в ЭВМ программ имеется одна, готовая к использованию освободившегося оборудования. Для уменьшения простоев оборудования ЭВМ широко применяют метод организации параллельной работы устройства ЭВМ за счет совмещения различных операций при работе ЭВМ. В целях более эффективного использования ЭВМ организуют мультипрограммную обработку информации на ЭВМ так, чтобы ею параллельно выполнялись команды, относящиеся к различным и независимым программам.

Мультипрограммный режим повышает производительность ЭВМ за счет увеличения числа задач, решаемых ЭВМ в течение некоторого промежутка времени. При этом время решения отдельной задачи увеличивается по сравнению с временем решения ее в однопрограммном режиме.

Режим пакетной обработки – это режим для обеспечения мультипрограммной обработки информации необходимо наличие нескольких задач, ожидающих обработки.

Для эффективной загрузки ЭВМ используется режим пакетной обработки данных. В этом режиме задачи (программы и данные), подготовляемые многими пользователями ЭВМ, собираются в пачки-пакеты. Пакет состоит из заданий (не более 15), относящимся ко многим задачам, обработка которых занимает не менее часа машинного времени.

Различают два режима пакетной обработки. В первом число задач, выполняемых одновременно, фиксируется, а во втором не фиксируется, но в процессе

обработки пакета ЭВМ оно может изменяться пакета ЭВМ оно может изменяться динамически. Пакет, предварительно записанный на том или ином носителе информации, вводится в ОЗУ ЭВМ. Когда пакет загружен, ЭВМ выбирает на обработку несколько задач и начинает выполнять их мультипрограммном режиме. Когда решение одной группы задач пакета закончено, из него выбирается для обработки следующая группа, это продолжается до тех пор, пока не будет обработана последняя группа задач пакета. После этого в ЭВМ вводится новый пакет задач.

Пакетная обработка данных позволяет увеличить производительность ЭВМ и уменьшить стоимость машинной обработки информации.

Режим разделения времени – предназначен для обеспечения непосредственного и одновременного доступа к ЭВМ некоторому количеству пользователей чаще всего с дистанционно удаленных пунктов (терминалов).

Терминал – периферийное устройство, предназначенное для обслуживания одного человека, решающего задачи на ЭВМ.

Пользователи с помощью терминалов вводят в ЭВМ исходные данные и программы и получают результаты вычислений. ЭВМ предоставляет каждому активному терминалу квант времени, равный секундам и долям секунды. По истечении этого времени ЭВМ переходит к обслуживанию следующего пользователя. За некоторый период времени ЭВМ обслуживает всех пользователей. При достаточно высоком быстродействии ЭВМ у отдельного пользователя создается иллюзия непрерывного контакта с ЭВМ.

Разделение времени позволяет устранить потери машинного времени, связанные с вмешательством оператора в работу ЭВМ из-за сравнительно низкой его скорости реакции, необходимости выполнения им определенных действий вне ЭВМ и медленного ввода информации с пульта оператора. При мультипрограммной работе ЭВМ в промежуточные паузы работы одного оператора к ЭВМ имеют доступ другие, что позволяет обеспечить полную загрузку внутренних устройств ЭВМ и тем самым поднять эффективность ее работы.

Режим разделения времени совместим с режимом пакетной обработки данных, которая предусматривается в ЭВМ для решения задач в отдельные периоды времени, когда пользователи не загружают ЭВМ полностью.

Диалоговый режим работы - это режим (режим "запрос-ответ"), при котором все программы пользователей постоянно хранятся в памяти ЭВМ и пользователи имеют непосредственный доступ к ЭВМ.

От пользователей в ЭВМ поступают входные данные и запросы с пультовых пишущих машинок или дисплеев. Ответ формируется по программе, соответствующей определенному запросу. Выбор допустимых запросов ограничен емкостью памяти. Каждый запрос имеет соответствующий приоритет и временные ограничения на срок обслуживания.

Режим работы в реальном масштабе времени - это режим, при котором ЭВМ управляет работой какого-либо объекта или технологического процесса.

Особенностью работы в реальном масштабе времени является то, что, помимо арифметической и логической обработки, выполняется слежение за работой объекта или прохождение некоторого процесса. Реализация этого режима привела к усложнению устройств и программного обеспечения ЭВМ.

1.4 Типы параллельной обработки информации

Эффективность использования вычислительных систем во многом определяется возможностями организации параллельной обработки. Параллелизм может возникать и обеспечиваться на разных уровнях подготовки и реализации вычислительного процесса – алгоритмов, программ, команд.

Под термином «параллельная обработка» мы будем понимать одновременное выполнение заданий, шагов (пунктов) заданий, программ, подпрограмм, циклов, операторов и команд.

Параллельная обработка информации может применяться с двумя основными целями:

- повышение производительности ЭВМ и ВС не за счет совершенствования элементной базы, а за счет эффективной организации вычислительных процессов;
- обеспечение высокой надежности ВС за счет дублирования вычислитель ной аппаратуры.

Повышение производительности ЭВМ и ВС — основная цель применения параллельной обработки, по этой причине параллельную архитектуру имеют такие ЭВМ, как многопроцессорные серверы, мэйнфреймы и супер-ЭВМ.

Параллельная обработка информации может производиться на нескольких уровнях (рис. 1.1).

Очевидно, что чем ниже уровень, тем мельче дробление программных процессов, тем мельче, как принято говорить, «зерно параллелизма». В общем случае возможно реализовать параллелизм как на отдельном уровне, так и на нескольких одновременно. Независимая однопроцессорная обработка реализует параллелизм на уровне 1. Векторная обработка заключается в параллельном выполнении циклов на уровне 2 и может производиться как на одном, так и нескольких процессорах. Уровни 3 и 4 соответствуют многопроцессорным ВС. Параллелизм уровня характерен для многомашинных вычислительных комплексов.

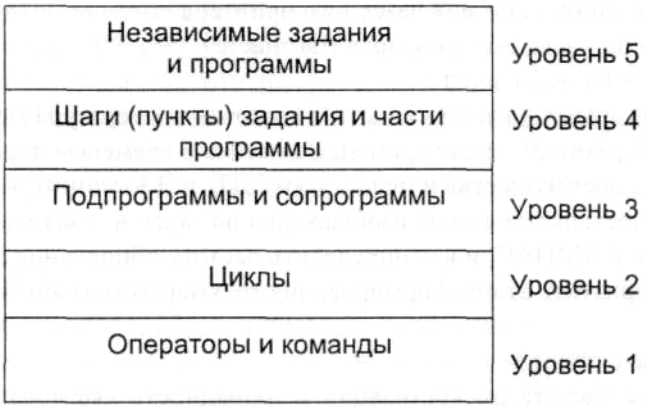


Рисунок 1.1 - Уровни параллелизма

Существуют два основных способа организации параллельной обработки:

- совмещение во времени этапов решения разных задач;
- одновременное решение различных задач или частей одной задачи.

Первый путь — совмещение во времени этапов решения разных задач — это *мультипрограммная обработка* информации. Мультипрограммная обработка уже давно и широко применяется для повышения производительности ЭВМ и ВС. Подробное рассмотрение мультипрограммной обработки относится к теме «Операционные системы» и выходит за рамки настоящего учебника.

Второй путь — одновременное решение различных задач или частей одной задачи — возможен только при наличии нескольких обрабатывающих устройств. При

этом используются те или иные особенности задач или потоков задач, что позволяет осуществить распараллеливание.

Можно выделить следующие типы параллелизма, позволяющие реализовать алгоритмические особенности отдельных задач и их потоков:

1) естественный параллелизм независимых задач;
2) параллелизм объектов или данных;
3) параллелизм ветвей задачи или программы.

Рассмотрим эти типы параллелизма.

1. *Естественный параллелизм* независимых задач заключается в том, что на вход ВС поступает непрерывный поток не связанных между собой задач, т. е. решение любой задачи не зависит от результатов решения других задач. В этом случае использование нескольких обрабатывающих устройств при любом способе комплексирования (объединения в систему) повышает производительность системы.

Характерным примером естественного параллелизма является поступление пользовательских запросов на информационный web-сайт. Каждый запрос порождает отдельную процедуру его исполнения, которая не зависит от других подобных процедур.

2. *Параллелизм объектов или данных* имеет место тогда, когда по одной и той же (или почти по одной и той же) программе должна обрабатываться некоторая совокупность данных, поступающих в систему одновременно.

Это могут быть, например, задачи обработки сигналов от радиолокационной станции: все сигналы обрабатываются по одной и той же программе. Другой пример — обработка информации от датчиков, измеряющих одновременно один и тот же параметр и Установленных на нескольких однотипных объектах.

Программы подобного типа могут быть различного объема и сложности, начиная от очень простых, содержащих несколько операций, до больших программ в сотни и тысячи операций. При этом параллельность выполнения операций достигается путем увеличения числа обрабатывающих устройств, каждое из которых способно автономно выполнять последовательность команд над отдельной совокупностью данных. Часто основной особенностью таких программ (в частности программ обработки векторов и матриц) является то, что одна и та же команда должна выполняться над большой совокупностью элементарных, связанных между

собой некоторым образом данных, и соответствующую операцию можно производить над всеми данными одновременно. При этом время решения задачи сокращается пропорционально числу обрабатывающих устройств.

3. *Параллелизм ветвей задачи или программы* — один из наиболее распространенных типов параллелизма в обработке информации. Он заключается в том что при решении одной задачи могут быть выделены отдельные ее части — ветви которые при наличии нескольких обрабатывающих устройств могут выполняться параллельно. При этом одновременно могут обрабатываться только независимые ветви задачи, т. е. такие ее части, для которых соблюдаются следующие условия:

- ни одна из выходных величин этих ветвей задачи не является входной величиной для другой такой ветви (отсутствие функциональных связей);
- условия выполнения одной ветви не зависят от результатов или признаков, полученных при выполнении других ветвей (независимость по управлению).

Хорошее представление о параллелизме ветвей дает ярусно-параллельная форма (ЯПФ) программы, пример которой приведен на рис. 1.2.

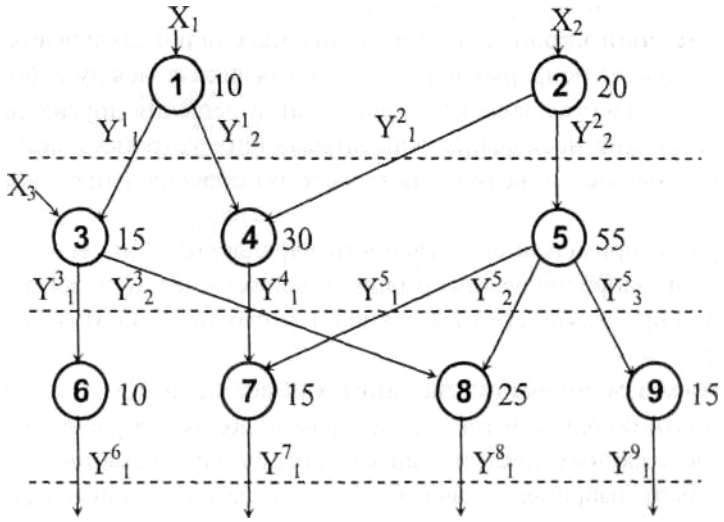


Рисунок 1.2 - Пример ярусно-параллельной формы программы

Программа представлена в виде совокупности ветвей, расположенных нескольких уровнях — ярусах. Кружками с цифрами внутри обозначены ветви Длина ветви представляется цифрой, стоящей около кружка и означающей, сколько временных единиц выполняется данная ветвь. Стрелками показаны входные данные и результаты обработки. Входные данные обозначаются символом X, выходные данные — символом Y. Символы X имеют нижние цифровые индексы, обозначающие номера входных величин; символы Y имеют цифровые индексы внизу, и вверху; цифра

вверху соответствует номеру ветви, при выполнении которой получен данный результат, а цифра внизу означает порядковый номер результата, полученного при реализации данной ветви программы. На одном ярусе размещаются независимые ветви задачи, не связанные друг с другом, т. е. результаты решения какой-либо ветви данного яруса не являются входными данными для другой ветви этого же яруса.

Изображенная на рис. 1.2 программа содержит 9 ветвей, расположенных на 3 ярусах. На примере этой, в общем, достаточно простой программы, можно выявить преимущества вычислительной системы, включающей несколько обрабатывающих устройств, и проблемы, которые при этом возникают.

Примем, что длина *i*-й ветви представляется числом временных единиц t_i, которые требуются для ее исполнения. Тогда нетрудно подсчитать, что для исполнения всей программы на 1 процессоре потребуется время T_1:

$$T_1 = \sum_{i=1}^{9}(10 + 20 + 15 + 30 + 55 + 10 + 15 + 25 + 15) = 195$$

Если представить, что программа выполняется двумя обрабатывающими устройствами (процессорами), работающими независимо друг от друга, то время решения задачи сократится. Однако это время, как нетрудно видеть, будет различным в зависимости от последовательности выполнения независимых ветвей.

Рассмотрим, например, такой вариант выполнения программы, представленной на рис. 5.2: пусть процессор 1 выполняет ветви 1-3-4-6-7-9, а процессор 2 выполняет ветви 2-5-8. На рис. 1.3 представлены временные диаграммы выполнения процессорами ветвей программы.

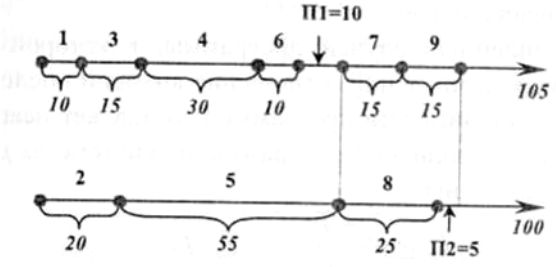


Рисунок 1.3 - Разложение ветвей программы по двум процессорам

Нетрудно подсчитать, что процессор 1 затрачивает 105, а процессор 2 — 100 единиц времени. При этом имеются два промежутка времени, когда один из процессоров вынужденно простаивает — *П1* длительностью 10 единиц и *П2*

длительностью 5 единиц времени. Промежуток *П1*, во время которого работает только процессор 2, образовался из-за того, что ветвь 7 зависит от ветви 5 (к моменту завершения ветви 6 еще не готовы данные Y_1^5). Промежуток *П1*, во время которого работает только процессор 1, образовался по причине окончания счета процессором 2.

Таким образом, на системе из двух процессоров наша программа будет выполнена полностью не менее, чем за 105 единиц времени. Величину, характеризующую уменьшение времени решения задачи на нескольких процессорах по сравнению с использованием одного процессора, называют ускорением счета *S* и рассчитывают как

$$S = \frac{T_1}{T_p} \quad , \qquad (1.1)$$

где T_p — время решения задачи на p процессорах. Кроме ускорения, используют такую величину, как коэффициент распараллеливания K_n, который рассчитывается как

$$K_n = \frac{T_1}{p * T_p} \quad . \qquad (1.2)$$

Коэффициент распараллеливания изменяется от 0 до 1 (от 0 до 100%) и отражает эффективность использования вычислительных ресурсов. В нашем примере нетрудно посчитать, что ускорение $S=195/105=1.86$, а коэффициент распараллеливания $K_n=0,93$. Как видим, по причине простоев одного из процессоров ускорение счета значительно меньше двух, т. е. количества используемых процессоров. Заметим, что в нашем примере не учитывались временные задержки, связанные с переключением контекстов программы (смены ветвей) и передачи данных от одной ветви к другой. Тем не менее в силу алгоритмических особенностей программы часть вычислений в промежутки *П1* и *П2* производится только одним процессором, т. е. фактически последовательно.

Рассмотрим обобщенный случай программы, в которой алгоритмически доля последовательных вычислений (отношение времени последовательных вычислений к общему времени счета программы) составляет некоторую величину *f*. В этом случае

время выполнения программы на системе из p процессоров не может быть меньше величины

$$T_p = \frac{T_1 - T_1 * f}{p} + T_1 * f \ , \qquad (1.3)$$

Первое слагаемое определяет время выполнения распараллеленной части программы, а второе слагаемое — время выполнения последовательной час программы. Нетрудно видеть, что ускорение счета в этом случае не будет превышать величины

$$S = \frac{1}{f + \frac{1-f}{p}} . \qquad (1.4)$$

Данное соотношение носит название *закона Амдала*[1] *(иногда закон Амдаля-Уэра)*. На примере программы рис. 1.3 можем видеть, что доля последовательных вычислений составляет *f=15/195*. Подставляя эту величину в формулу закона Амдала, получаем для системы из двух процессоров максимальное ускорение 1,86 раза, что соответствует ранее рассчитанному значению.

Для иллюстрации действия закона Амдала приведем следующий пример. Пусть доля последовательных вычислений в некоторой программе составляет 10%. Тогда максимальное ускорение счета на 100 процессорах не превысит 9,2. Коэффициент распараллеливания составит всего лишь 9,2%. На 10 процессорах ускорение составит 5,3, а коэффициент распараллеливания — 53%. Нетрудно видеть, что даже такая небольшая доля последовательных вычислений уже на теоретическом уровне, без учета неизбежных задержек в реальной ВС, серьезно ограничивает возможности масштабирования программы.

---

[1] Джим Амдал (Eugene Amdahl) родился 16 ноября 1922 года, величайший проектировщик компьютерных систем XX века. Он является конструктором и разработчиком таких легендарных компьютеров, как IBM 704, 709, 7030, 7090 и архитектором компьютерного семейства третьего поколения IBM/360. Последнее начинание (1994 год) Джина Амдала — компания Commercial Data Servers (CDS). Она разрабатывает недорогие мэйнфреймы для строго определенной группы покупателей. Первой разработкой компании CDS стал небольшой мэйнфрейм CDS 104 с производительностью 7 млн. операций в секунду. “Величайший” проектировщик мэйнфреймов продолжает работать, несмотря на свой почтенный возраст. Продолжает действовать и его закон — закон Амдала, выведенный им еще в конце 60-х годов XX века.

Определим, какая должна быть максимальная доля / последовательных вычислений в программе, чтобы было возможно получить наперед заданное ускорение счета S с максимальным коэффициентом распараллеливания K_n. Для этого выразим из закона Амдала долю последовательных вычислений:

$$f \le \frac{p - S}{S * (p - 1)}, \qquad (1.5)$$

Из соотношения (1.5) видно, что для того, чтобы допустить наличие последовательных вычислений в программе (f должно быть больше 0), число процессоров p должно быть больше предполагаемого ускорения S. При этом условии максимальный коэффициент распараллеливания K_n получается при числе процессоров $p=S+1$.

Таким образом, доля последовательных вычислений должна составлять не более

$$f \le \frac{1}{S^2}. \qquad (1.6)$$

Соотношение (1.6) определяет очень важное следствие из закона Амдала. Для того чтобы ускорить программу в q раз, необходимо ускорить не менее, чем $\left(1 - \frac{1}{q^2}\right)$-ю часть программы. Например, чтобы получить ускорение в 100 раз, необходимо распараллелить 99,99% всей программы.

Кроме алгоритмического распараллеливания, для того чтобы с помощью нескольких обрабатывающих устройств решить задачу, имеющую параллельные ветви, необходима соответствующая организация процесса, которая определяет пути решения задачи и вырабатывает необходимую информацию о готовности каждой ветви. Однако все это относительно легко реализовать тогда, когда известна достаточно точно длительность выполнения каждой ветви. На практике это бывает крайне редко: в лучшем случае имеется та или иная временная оценка. Поэтому организация оптимального или близкого к оптимальному графика работы является достаточно сложной задачей.

Следует отметить также и определенные сложности, связанные с выделением независимых ветвей при разработке программ. Вместе с тем при решении многих сложных задач только программирование с выделением независимых ветвей позволяет существенно сократить время решения. В частности, хорошо поддаются параллельной обработке такого типа задачи матричной алгебры, линейного программирования, спектральной обработки сигналов, прямые и обратные преобразования Фурье и др.

1.5 Классификация архитектуры ВС с параллельной обработкой данных

1.5.1 Классификация М. Флинна[1]

Цели, которым должна служить хорошо построенная классификация архитектур:

- облегчать понимание того, что достигнуто на сегодняшний день в области архитектур вычислительных систем, и какие архитектуры имеют лучшие перспективы в будущем;
- подсказывать новые пути организации архитектур — речь идет о тех классах, которые в настоящее время по разным причинам пусты;
- показывать, за счет каких структурных особенностей достигается увеличение производительности различных вычислительных систем; с этой точки зрения классификация может служить моделью для анализа производительности.

---

[1] Майкл Дж. Флинн (Michael J. Flynn) - работал в IBM (1955 - 1965 г). Руководитель проектов IBM 7090 и 7094/II, а также System 360 Model 91 Central Processing Unit. В 1975 году он стал профессором электротехники в Стэнфордском университете, директор лаборатории компьютерных систем (1977-1983 г.). Был основателем и председателем ACM Special Interest Group по компьютерной архитектуре и IEEE Computer Society (технический комитет по компьютерной архитектуре). Награжден премией за технический вклад в компьютерную и цифровую архитектуру систем и мемориальной премией в знак признания его выдающегося вклада в разработку и классификацию архитектуры вычислительных систем. Является автором трех книг и более 250 технических документов.

Таблица 1.1 – Классификация Флинна

Поток данных	Поток команд	
	одиночный	**множественный**
одиночный	SISD – Single Instruction stream/ Single Data stream (одиночный поток команд и одиночный поток данных – ОКОД)	MISD – Multiple Instruction stream/ Single Data stream (множественный поток команд и одиночный поток данных – МКОД)
множественный	SIMD – Single Instruction stream/ Multiple Data stream (одиночный поток команд и множественный поток данных – ОКМД)	MIMD – Multiple Instruction stream/ Multiple Data stream (множественный поток команд и множественный поток данных – ОКМД)

В 1966 г. М. Флинном был предложен следующий подход к классификации архитектур вычислительных систем. В основу было положено понятие потока, под которым понимается последовательность элементов, команд или данных, обрабатываемая процессором. Соответствующая система классификации основана на рассмотрении числа потоков инструкций и потоков данных и описывает четыре базовых класса (табл. 1.1, рис. 1.4).

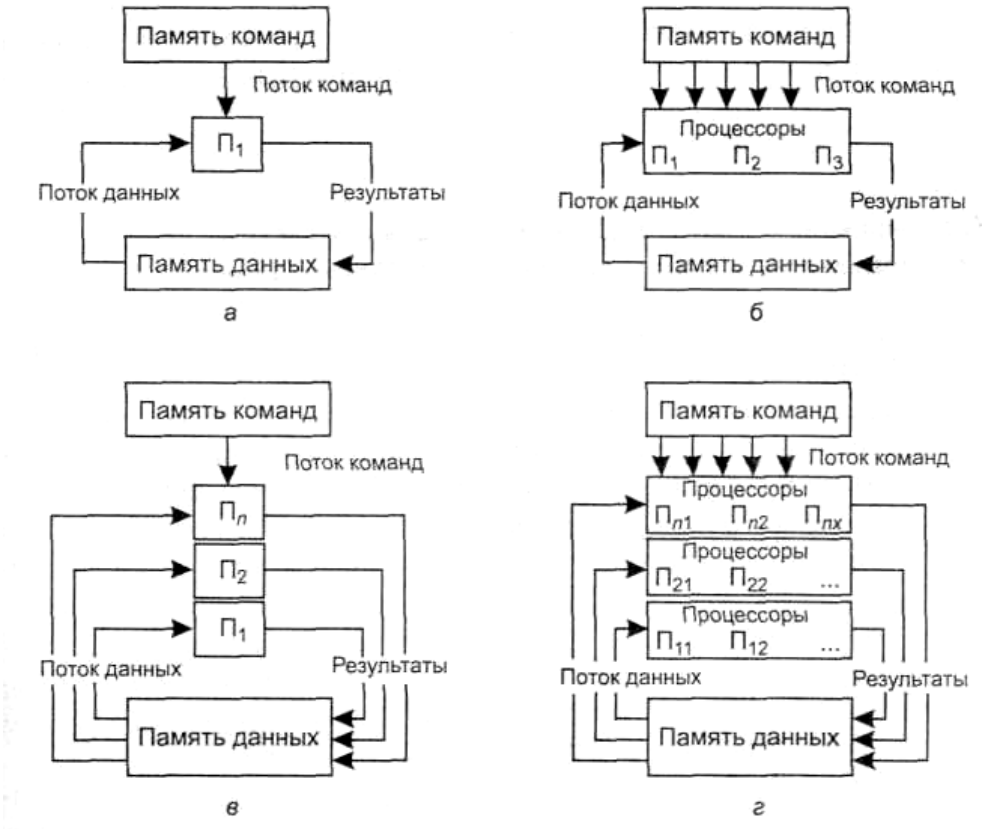


Рисунок 1.4 - Классификация Флинна:

***а* - SISD; *б* - MISD; *в* - SIMD; *г* - MIMD**

Коротко рассмотрим отличительные особенности каждой из архитектур.

Архитектура ОКОД (SISD) охватывает все однопроцессорные и одномашинные варианты систем, т. е. с одним вычислителем. Все ЭВМ классической структуры попадают в этот класс. Здесь параллелизм вычислений обеспечивается путем совмещения выполнения операций отдельными блоками АЛУ, а также параллельной работой устройств ввода-вывода информации и процессора. Закономерности организации вычислительного процесса в этих структурах достаточно хорошо изучены.

Архитектура ОКМД (SIMD) предполагает создание структур век торной или матричной обработки. Системы этого типа обычно строятся как однородные, т. е. процессорные элементы, входящие в систему, идентичны, и все они управляются одной и той же последовательностью команд. Однако каждый процессор обрабатывает свой поток данных. Под эту схему хорошо подходят задачи обработки матриц или векторов (массивов), задачи решения систем линейных и нелинейных, алгебраических и дифференциальных уравнений, задачи теории поля и др. В структурах данной архитектуры желательно обеспечивать соединения между процессорами, соответствующие реализуемым математическим зависимостям. Как правило, эти связи напоминают матрицу, в которой каждый процессорный элемент связан с соседними. По данной схеме строились системы: первая суперЭВМ — ILLIAC-IV, отечественные параллельные системы — ПС-2000, ПС-3000. Идея векторной обработки широко использовалась в таких известных суперЭВМ, как Cyber-205 и Gray-I, II, III. Узким местом подобных систем является необходимость изменения коммутации между процессорами, когда связь между ними отличается от матричной. Кроме того, класс задач, допускающих широкий матричный параллелизм, весьма узок. Структуры ВС этого типа, по существу, являются структурами специализированных суперЭВМ.

Элементы технологии SIMD реализованы в процессорах Intel начиная с Pentium MMX (1997 г.).

Архитектура — МКОД (MISD) предполагает построение своеобразного процессорного конвейера, в котором результаты обработки передаются от одного процессора к другому по цепочке. Выгоды такого вида обработки понятны. Прототипом таких вычислений может служить схема любого производственного

конвейера. В современных ЭВМ по этому принципу реализована схема совмещения операций, в которой параллельно работают раз личные функциональные блоки, и каждый из них делает свою часть в общем цикле обработки команды. В ВС этого типа конвейеры должны образовывать группы процессоров. Однако при переходе на системный уровень очень трудно выявить подобный регулярный характер в универсальных вычислениях. Кроме того, на практике нельзя обеспечить и «большую длину» такого конвейера, при которой достигается наивысший эффект. Вместе с тем конвейерная схема нашла применение в так называемых скалярных процессорах суперЭВМ, в которых они применяются как специальные процессоры для поддержки векторной обработки.

Архитектура МКМД (MIMD) предполагает, что все процессоры системы работают по своим программам с собственным потоком команд. В простейшем случае они могут быть автономны и независимы. Такая схема использования ВС часто применяется на многих крупных вычислительных центрах для увеличения пропускной способности центра. Большой интерес представляет возможность согласованной работы ЭВМ (процессоров), когда каждый элемент делает часть общей задачи. Общая теоретическая база такого вида работ практически отсутствует. Но можно привести примеры большой эффективности этой модели вычислений. Подобные системы могут быть многомашинными и многопроцессорными. Например, отечественный проект машины динамической архитектуры (МДА) — ЕС-2704, ЕС-2727 — предполагал одновременное использование сотни процессоров.

1.5.2 Другие подходы к классификации ВС

Наличие большого разнообразия систем, образующих класс МКМД (MIMD), делает классификацию Флинна не полностью адекватной. Действительно и 4-процессорный SX-5 компании NEC и 1000-процессорный Cray T3E попадают в этот класс, и это заставляет искать другие подходы к классификации.

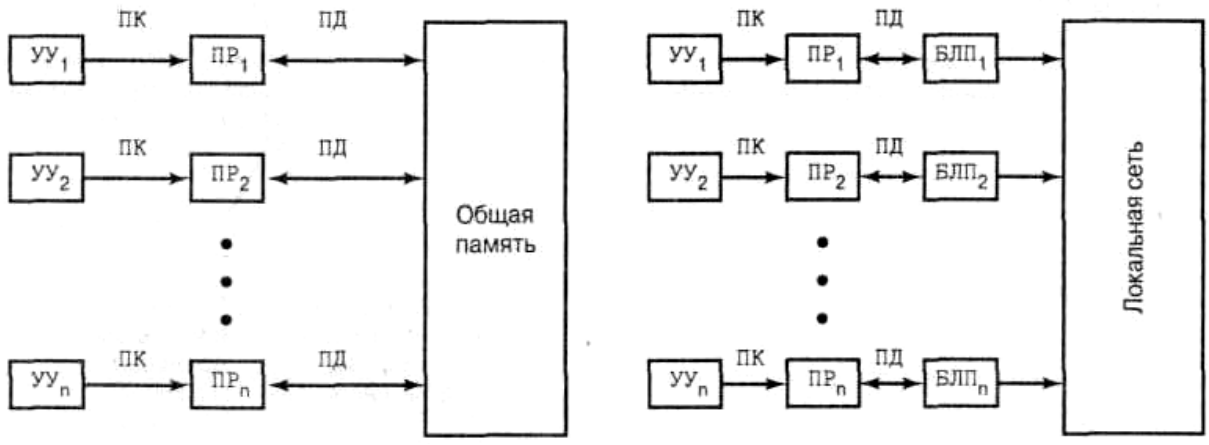
Классификация Е. Джонсона. Джонсон предложил проводить классификацию MIMD-архитектур на основе структуры памяти и реализации механизма взаимодействия и синхронизации между процессорами.

По структуре оперативной памяти существующие вычислительные системы делятся на две большие группы: либо это системы с общей памятью, прямо адресуемой всеми

процессорами, либо это системы с распределенной памятью, каждая часть которой доступна только одному процессору. Одновременно с этим и для межпроцессорного взаимодействия существуют две альтернативы: через разделяемые (общие) переменные или с помощью механизма передачи сообщений. Исходя из таких предположений, можно получить четыре класса MIMD-архитектур, уточняющих систематику Флинна (табл. 1.2).

Таблица 1.2 - Классификация Джонсона для систем MIMD по Флинну

Обмен	**Память**	
	общая	**распределенная**
общие данные	GMSV - General Memory-Shared variables (Общая память - разделяемые переменные) Класс 1. «Системы с разделяемой памятью»	DMSV - Distributed Memory, Shared variables (Распределенная память - разделяемые переменные) Класс 2. «Гибридная архитектура»
передача данных	GMMP - General Memory, Message propagation (Общая память - передача сообщений)	DMMP - Distributed Memory, Message propagation) (Распределенная память - передача сообщений) Класс 3. «Архитектуры с передачей сообщений»



а

УУ – устройство управления
ПР – процессор
ПД – поток данных
ПК - поток команд
БЛП – блок локальной памяти

б

Рисунок 1.5 - Структуры: а) с общей памятью; б) с распределенной памятью

Основываясь на таком делении, Джонсон вводит следующие наименования для некоторых классов:

- вычислительные системы, использующие общую разделяемую память для межпроцессорного взаимодействия и синхронизации, он называет системами с разделяемой памятью, например CRAY Y-MP (по его классификации это класс 1);
- системы, в которых память распределена по процессорам, а для взаимодействия и синхронизации используется механизм передачи сообщений, называются архитектурами с передачей сообщений, например NCube (класс 3);
- системы с распределенной памятью и синхронизацией через разделяемые переменные, как в BBN Butterfly, называются гибридными архитектурами (класс 2).

В качестве уточнения классификации автор отмечает возможность учитывать вид связи между процессорами: общая шина, переключатели, разнообразные сети и т. п.

Классификация Базу. По мнению А. Базу (A. Basu), любую параллельную вычислительную систему можно однозначно описать последовательностью решений, принятых на этапе ее проектирования, а сам процесс проектирования представить в виде дерева. Корень дерева — это вычислительная система (рис. 1.6), и последующие ярусы дерева, фиксируя уровень параллелизма, метод реализации алгоритма, параллелизм инструкций и способ управления, последовательно дополняют друг друга, формируя описание системы.

На первом этапе определяется, какой уровень параллелизма использует вычислительная система. Одна и та же операция может одновременно выполняться над целым набором данных, определяя параллелизм на уровне данных (обозначено *D* на рис. 1.6). Способность выполнять более одной операции одновременно говорит о параллелизме на уровне команд *(O)*. Если же компьютер спроектирован так, что целые последовательности команд могут быть выполнены одновременно, то будем говорить о параллелизме на уровне задач *(T)*.

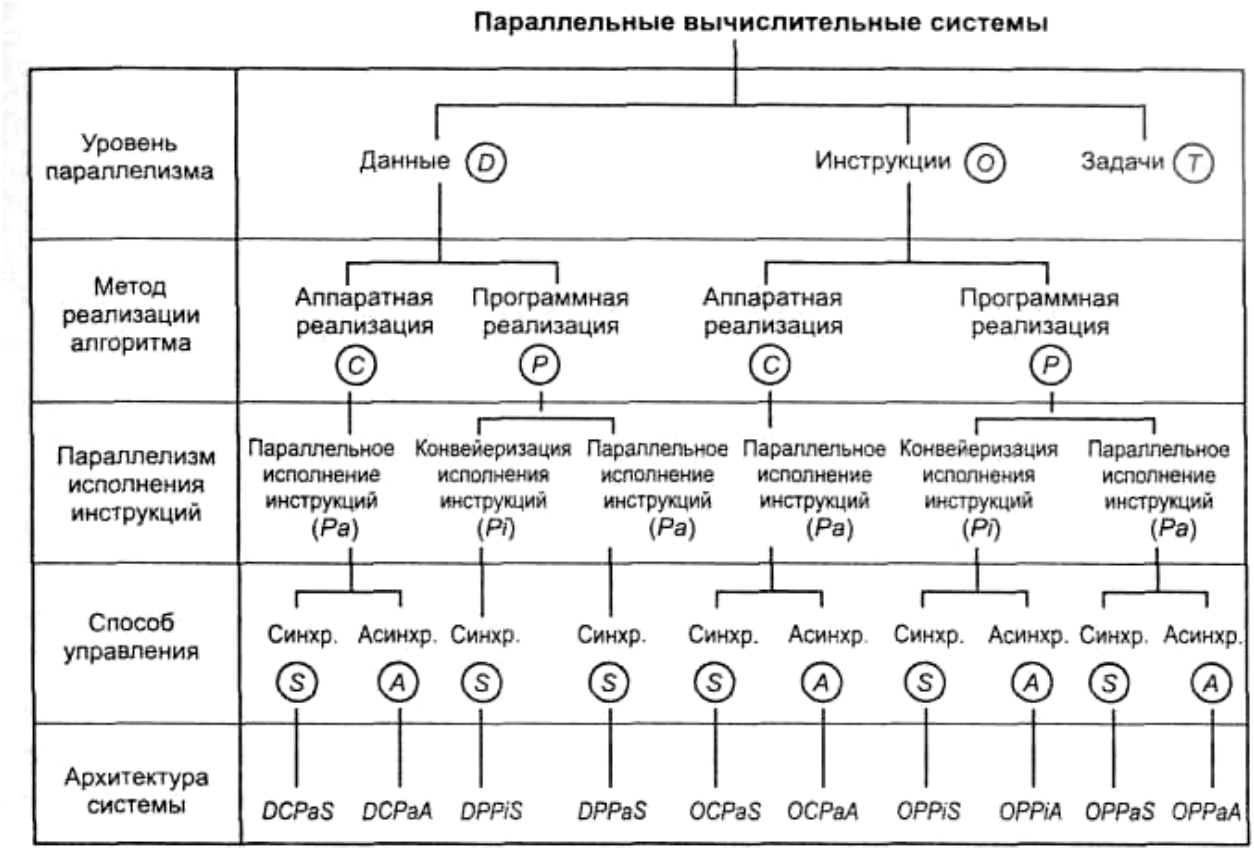


Рисунок 1.6 - Классификация Базу

Второй уровень в классификационном дереве фиксирует метод реализации алгоритма. С появлением сверхбольших интегральных схем (СБИС) стало возможным реализовывать аппаратно не только простые арифметические операции, но и алгоритмы целиком. Например, быстрое преобразование Фурье, перемножение матриц и другие относятся к классу тех алгоритмов, которые могут быть эффективно реализованы в СБИС. Данный уровень классификации разделяет системы с аппаратной реализацией алгоритмов (*C* на рис. 1.6) и системы, использующие традиционный способ программной реализации *(P)*.

Третий уровень конкретизирует тип параллелизма, используемого для обработки инструкций машины: конвейеризация инструкций (P_i) или их независимое (параллельное) выполнение *(*P_a*)*. В большей степени этот выбор относится к компьютерам с программной реализацией алгоритмов, так как аппаратная реализация всегда предполагает параллельное исполнение команд. Отметим, что в случае конвейерного исполнения имеется в виду лишь конвейеризация самих команд, разбивающая весь цикл обработки на выборку команды, дешифрацию, вычисление

адресов и т. д. (возможная конвейеризация вычислений на данном уровне не принимается во внимание).

Последний уровень данной классификации определяет способ управления, принятый в вычислительной системе: синхронный *(S)* или асинхронный *(A)*. Если выполнение команд происходит в строгом порядке, определяемом только сигналами таймера и счетчиком команд, то говорят о синхронном способе управления. Если же для инициации команды определяющими являются такие факторы, как, например, готовность данных, то машина попадает в класс с асинхронным управлением. Наиболее характерными представителями систем с асинхронным управлением являются компьютеры *data-driven* и *demand-driven*.

Рисунок 1.7 - Классификация Дункана

Классификация Р. Дункана. Р. Дункан определяет тот набор требований (рис. 1.7), на который может опираться искомая классификация.

Из класса параллельных машин должны быть исключены те, в которых параллелизм заложен лишь на самом низком уровне, включая:

- конвейеризацию на этапе подготовки и выполнения команды (instruction pipelining), т. е. частичное перекрытие таких этапов, как дешифрация команды, вычисление адресов операндов, выборка операндов, выполнение команды и сохранение результата;
- наличие в архитектуре нескольких функциональных устройств,

работающих независимо, в частности, возможность параллельного выполнения логических и арифметических операций;

- наличие отдельных процессоров ввода/вывода, работающих независимо и параллельно с основными процессорами.

Причины исключения перечисленных выше особенностей автор объясняет следующим образом. Если рассматривать компьютеры, использующие только параллелизм низкого уровня, наравне со всеми остальными, то, во-первых, практически все существующие системы будут классифицированы как «параллельные» (что заведомо не будет позитивным фактором для классификации), и, во-вторых, такие машины будут плохо вписываться в любую модель или концепцию, отражающую параллелизм высокого уровня.

Классификация должна быть согласованной с классификацией Флинна, показавшей правильность выбора идеи потоков команд и данных.

Классификация должна описывать архитектуры, которые однозначно не укладываются в систематику Флинна, но тем не менее относятся к параллельным архитектурам (например, векторно-конвейерные).

Учитывая вышеизложенные требования, Дункан дает неформальное определение параллельной архитектуры, причем именно неформальность дала ему возможность включить в данный класс компьютеры, которые ранее не вписывались в систематику Флинна. Итак, *параллельная архитектура — это такой способ организации вычислительной системы, при котором допускается, чтобы множество процессоров (простых или сложных) могло бы работать одновременно, взаимодействуя по мере надобности друг с другом.* Следуя этому определению, все разнообразие параллельных архитектур Дункан систематизирует так, как это показано на рис. 1.7.

По существу систематика очень простая: процессоры системы работают либо синхронно, либо независимо друг от друга, либо в архитектуру системы заложена та или иная модификация идеи MIMD. На следующем уровне происходит детализация в рамках каждого из этих трех классов. Дадим небольшое пояснение лишь к тем из них, которые на сегодняшний день не столь широко известны.

Систолические архитектуры (их чаще называют систолическими массивами) представляют собой множество процессоров, объединенных регулярным образом (например, система WARP). Обращение к памяти может осуществляться только через определенные процессоры на границе массива. Выборка операндов из памяти и передача данных по массиву осуществляется в одном и том же темпе. Направление передачи данных между процессорами фиксированно. Каждый процессор за интервал времени выполняет небольшую инвариантную последовательность действий.

Гибридные MIMD/SIMD архитектуры, вычислительные системы *dataflow, reduction* и *wavefront* осуществляют параллельную обработку информации на основе асинхронного управления, как и MIMD-системы. Но они выделены в отдельную группу, поскольку все имеют ряд специфических особенностей, которыми не обладают системы, традиционно относящиеся к MIMD.

MIMD/SIMD — типично гибридная архитектура. Она предполагает, что в MIMD-системе можно выделить группу процессоров, представляющую собой подсистему, работающую в режиме SIMD (например, PASM, Non-Von). Такие системы отличаются относительной гибкостью, поскольку допускают реконфигурацию в соответствии с особенностями решаемой прикладной задачи.

Остальные три вида архитектур используют нетрадиционные модели вычислений. *Dataflow-машины* используют модель, в которой команда может выполняться сразу же, как только вычислены необходимые операнды. Таким образом, последовательность выполнения команд определяется зависимостью по данным, которая может быть выражена, например, в форме графа.

Модель вычислений, применяемая в *reduction-машинах*, иная и состоит в следующем: команда становится доступной для выполнения тогда и только тогда, когда результат ее работы требуется другой, доступной для выполнения команде в качестве операнда.

Архитектура *wavefront array* объединяет в себе идею систолической обработки данных и модель вычислений, используемой в dataflow-машинах. В данной архитектуре процессоры объединяются в модули и связи, по которым процессоры могут взаимодействовать друг с другом, фиксируются. Однако, в противоположность ритмичной работе систолических массивов, данная архитектура использует асинхронный

механизм связи с подтверждением *(handshaking),* из-за чего «фронт волны» вычислений может менять свою форму по мере прохождения по всему множеству процессоров.

Классификация Е. Кришнамарфи. Кришнамарфи для классификации параллельных вычислительных систем предлагает использовать четыре характеристики, похожие на характеристики классификации А. Базу (рис. 1.8):

- степень гранулярности;
- способ реализации параллелизма;
- топологию и природу связи процессоров;
- способ управления процессорами.

Принцип построения классификации достаточно прост. Для каждой степени гранулярности рассматриваются все возможные способы реализации параллелизма. Для каждого полученного таким образом варианта устанавливаются все комбинации топологии связи и способов управления процессорами. В результате получается дерево (см. рис. 1.8), в котором каждый ярус соответствует своей характеристике, каждый лист представляет отдельную группу компьютеров в данной классификации, а путь от вершины дерева однозначно определяет значения указанных выше характеристик.

Рисунок 1.8 - Классификация Кришнамарфи

Первых два уровня практически повторяют классификацию А. Базу. Третий уровень классификации (топология и природа связи процессоров) тесно связан со

вторым. Если был выбран аппаратный способ реализации параллелизма, то надо рассмотреть топологию связи процессоров (матрица, линейный массив, тор, дерево, звезда и т. п.) и степень связности процессоров между собой (сильная, слабая или средняя), которая определяется относительной долей накладных расходов при организации взаимодействия процессоров. В случае комбинированной реализации параллелизма, помимо топологии и степени связности, надо дополнительно учесть механизм взаимодействия процессоров: передача сообщений, разделяемые переменные или принцип dataflow (по готовности операндов).

Наконец, последний, четвертый уровень — способ управления процессорами, определяет общий принцип функционирования всей совокупности процессоров вычислительной системы: синхронный, dataflow или асинхронный.

На основе выделенных четырех характеристик нетрудно определить место наиболее известных классов архитектур в данной систематике.

Векторно-конвейерные компьютеры:

- гранулярность — на уровне данных;
- реализация параллелизма — аппаратная;
- связь процессоров — простая топология со средней связностью;
- способ управления — синхронный.

Классические мультипроцессоры:

- гранулярность — на уровне задач
- реализация параллелизма — комбинированная;
- связь процессоров — простая топология со слабой связностью и использованием разделяемых переменных;
- способ управления — асинхронный.

Матричные процессоры:

- гранулярность — на уровне данных;
- реализация параллелизма — аппаратная;
- связь процессоров — двухмерные массивы с сильной связностью;
- способ управления — синхронный.

Систолические массивы:

- гранулярность — на уровне данных;
- реализация параллелизма — аппаратная;

- связь процессоров — сложная топология с сильной связностью;
- способ управления — синхронный.

Архитектура типа wavefront:

- гранулярность — на уровне данных;
- реализация параллелизма — аппаратная;
- связь процессоров — двухмерная топология с сильной связностью;
- способ управления — dataflow.

Архитектура типа dataflow:

- гранулярность — на уровне команд;
- реализация параллелизма — комбинированная;
- связь процессоров — простая топология с сильной либо средней связностью и использованием принципа dataflow;
- способ управления — асинхронно-dataflow.

Несмотря на то, что классификация Е. Кришнамарфи построена только на четырех признаках, она позволяет выделить и описать такие «нетрадиционные» параллельные системы, как систолические массивы, машины типа dataflow и wavefront. Однако эта же простота является и основной причиной ее недостатков: некоторые архитектуры нельзя однозначно отнести к тому или иному классу, например, компьютеры с архитектурой гиперкуба и ассоциативные процессоры. Для более точного описания таких машин потребуется ввести еще целый ряд характеристик, таких, как размещение задач по процессорам, способ маршрутизации сообщений, возможность реконфигурации, аппаратная поддержка языков программирования и другие. Вместе с тем ясно, что эти признаки формализовать гораздо труднее, поэтому есть опасность вместо ясности внести в описание лишь дополнительные трудности.

Классификация Д. Скилликорна. Скилликорн разработал подход, ориентированный как на описание свойств многопроцессорных систем, так и некоторых нетрадиционных архитектур, в частности dataflow и reduction-машины.

Предлагается рассматривать архитектуру любого компьютера, как абстрактную структуру, состоящую из четырех компонент:

- *процессор команд* (*IP — Instruction Processor*) — функциональное устройство, работающее, как интерпретатор команд; в системе, вообще говоря, может отсутствовать;

- *процессор данных* (*DP — Data Processor*) — функциональное устройство, работающее как преобразователь данных, в соответствии с арифметическими операциями;
- *иерархия памяти* (IM — Instruction Memory, DM — Data Memory) — запоминающее устройство, в котором хранятся данные и команды, пересылаемые между процессорами;
- *переключатель — абстрактное устройство, обеспечивающее связь между процессорами и памятью.*

Функции процессора команд во многом схожи с функциями устройств управления последовательных машин и, согласно Д. Скилликорну, сводятся к следующим:

- на основе своего состояния и полученной от DP информации IP определяет адрес команды, которая будет выполняться следующей;
- осуществляет доступ к IM для выборки команды;
- получает и декодирует выбранную команду;
- сообщает DP команду, которую надо выполнить;
- определяет адреса операндов и посылает их в DP;
- получает от DP информацию о результате выполнения команды.

Функции процессора данных делают его во многом похожим на арифметическое устройство традиционных процессоров:

- DP получает от IP команду, которую надо выполнить;
- получает от IP адреса операндов;
- выбирает операнды из DM;
- выполняет команду;
- запоминает результат в DM;
- возвращает в IP информацию о состоянии после выполнения команды.

В терминах, таким образом, определенных основных частей компьютера структуру традиционной фон-неймановской архитектуры можно представить в следующем виде (рис. 1.9).

Рисунок 1.9 - Представление фон-неймановской архитектуры по Скилликорну

Это один из самых простых видов архитектуры, не содержащих переключателей. Для описания параллельных вычислительных систем автор зафиксировал четыре типа переключателей, без какой-либо явной связи с типом устройств, которые они соединяют:

- *1-1* — переключатель такого типа связывает пару функциональных устройств;
- *n-n* — переключатель связывает *i*-е устройство из одного множества устройств с *i*-м устройством из другого множества, т. е. фиксирует попарную связь;
- *1-n* — переключатель соединяет одно выделенное устройство со всеми функциональными устройствами из некоторого набора;
- *nxn* — каждое функциональное устройство одного множества может быть связано с любым устройством другого множества, и наоборот.

Примеров подобных переключателей можно привести очень много. Так, все матричные процессоры имеют переключатель типа 1-я для связи единственного процессора команд со всеми процессорами данных. В компьютерах семейства Connection Machine каждый процессор данных имеет свою локальную память, следовательно, связь будет описываться как *n-n*. В то же время, каждый процессор команд может связаться с любым другим процессором, поэтому данная связь будет описана как *nxn*.

Классификация Д. Скилликорна состоит из двух уровней. На первом уровне она проводится на основе восьми характеристик:

количества процессоров команд (IP);

- числа запоминающих устройств (модулей памяти) команд (IM);
- типа переключателя между IP и IM;
- количества процессоров данных (DP);
- числа запоминающих устройств (модулей памяти) данных (DM);
- типа переключателя между DP и DM;
- типа переключателя между IP и DP;
- типа переключателя между DP и DP.

Используя введенные характеристики и предполагая, что рассмотрение количественных характеристик можно ограничить только тремя возможными вариантами значений: 0, 1 и л (т. е. больше одного), можно получить 28 классов архитектур.

В классах 1—5 находятся компьютеры типа dataflow и reduction, не имеющие процессоров команд в обычном понимании этого слова. Класс 6 — это классическая фон-неймановская последовательная машина. Все разновидности матричных процессоров содержатся в классах 7—10. Классы 11 и 12 отвечают компьютерам типа MISD классификации Флинна и на настоящий момент, по мнению автора, пусты. Классы с 13-го по 28-й занимают всесозможные варианты мультипроцессоров, причем в 13—20 классах находятся машины с достаточно привычной архитектурой, в то время, как архитектура классов 21—28 пока выглядит экзотично.

На втором уровне классификации Д. Скилликорн уточняет описание, сделанное на первом уровне, добавляя возможность конвейерной обработки в процессорах команд и данных.

Классификация В. Хендлера. В основу классификации Хендлер закладывает явное описание возможностей параллельной и конвейерной обработки информации вычислительной системой. При этом он намеренно не рассматривает различные способы связи между процессорами и блоками памяти и считает, что

коммуникационная сеть может быть нужным образом сконфигурирована и будет способна выдержать предполагаемую нагрузку (рис. 1.10).

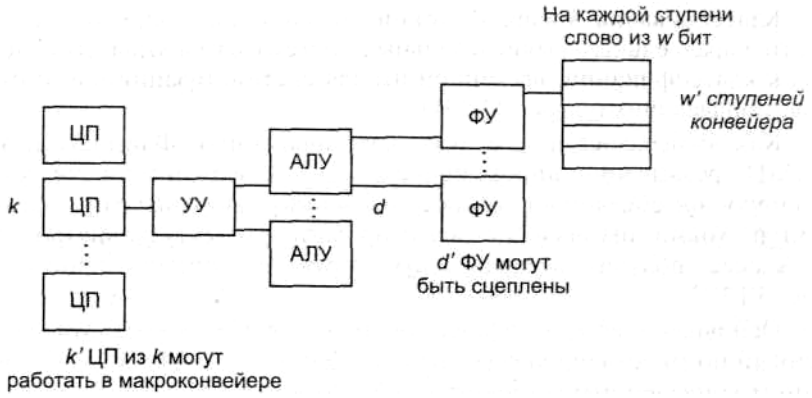


Рисунок 1.10 - Классификация Хендлера

Предложенная классификация базируется на различии между тремя уровнями обработки данных в процессе выполнения программ:

- уровень выполнения программы: опираясь на счетчик команд и некоторые другие регистры, устройство управления (УУ) производит выборку и дешифрацию команд программы;
- уровень выполнения команд: арифметико-логическое устройство компьютера (АЛУ) исполняет команду, выданную ему устройством управления;
- уровень битовой обработки: все элементарные логические схемы процессора (ЭЛС) разбиваются на группы, необходимые для выполнения операций над одним двоичным разрядом.

Таким образом, подобная схема выделения уровней предполагает, что вычислительная система включает какое-то число процессоров, каждый со своим устройством управления. Каждое устройство управления связано с несколькими арифметико-логическими устройствами, исполняющими одну и ту же операцию в каждый конкретный момент времени. Наконец, каждое АЛУ объединяет несколько элементарных логических схем, ассоциированных с обработкой одного двоичного разряда (число ЭЛС есть не что иное, как длина машинного слова). Если на какое-то время не рассматривать возможность конвейеризации, то число устройств управления κ, число арифметико-логических устройств d в каждом устройстве управления и

число элементарных логических схем *w* в каждом АЛУ составят тройку для описания данной вычислительной системы *C*:

$$t(C) = (k, d, w).$$

Классификация Р. Хокни[1]. Хокни разработал свой подход к классификации, введенной им для систематизации компьютеров, попадающих в класс MIMD.

Основная идея классификации (см. рис. 1.11) состоит в следующем. Множественный поток команд может быть обработан двумя способами: либо одним конвейерным устройством обработки, работающем в режиме разделения времени для отдельных потоков, либо каждый поток обрабатывается своим собственным устройством. Первая возможность используется в MIMD-компьютерах, которые автор называет конвейерными (например, процессорные модули в Denelcor HEP). Архитектуры, использующие вторую возможность, в свою очередь опять делятся на два класса:

- MIMD-компьютеры, в которых возможна прямая связь каждой пары процессоров, которая реализуется с помощью переключателя;
- MIMD-компьютеры, в которых прямая связь каждого процессора возможна только с ближайшими соседями по сети, а взаимодействие удаленных процессоров поддерживается специальной системой маршрутизации через процессоры-посредники.

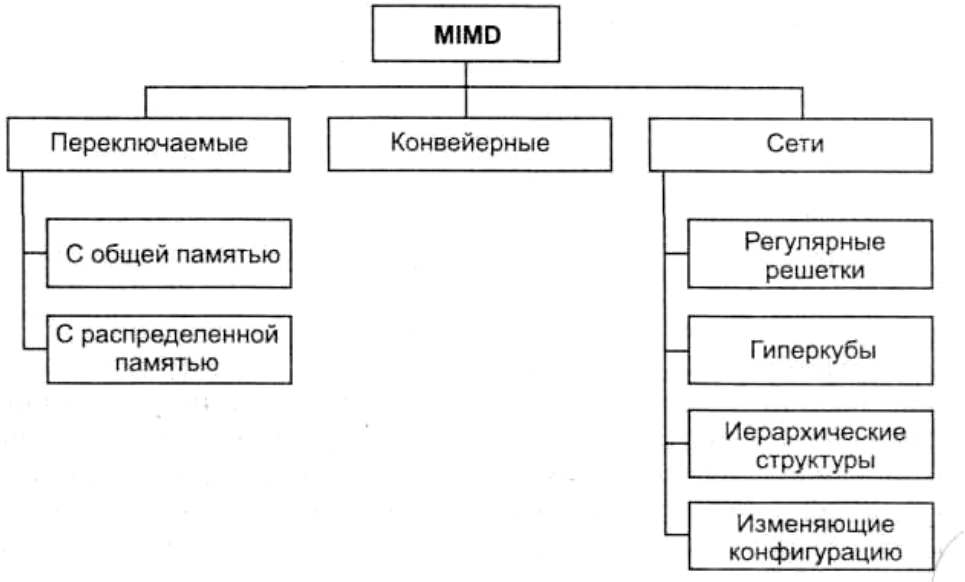


Рисунок 1.11 - Системы архитектуры MIMD (Флинн) в интерпретации Хокни

[1] Р. Хокни - английский специалист в области параллельных вычислительных систем

Далее, среди MIMD-машин с переключателем Хокни выделяет те, в которых вся память распределена среди процессоров как их локальная память (например, PASM, PRINGLE). В этом случае общение самих процессоров реализуется с помощью очень сложного переключателя, составляющего значительную часть компьютера. Такие машины носят название MIMD-машин *с распределенной памятью*. Если память — это разделяемый ресурс, доступный всем процессорам через переключатель, то такие MlMD-машины являются системами *с общей памятью* (CRAY X-MP, BBN Butterfly). В соответствии с типом переключателей можно проводить классификацию и далее: простой переключатель, многокаскадный переключатель, обшая шина.

Многие современные вычислительные системы имеют как общую разделяемую память, так и распределенную локальную. Такие системы являются *гибридными* MIMD с переключателем.

При рассмотрении MIMD-машин с *сетевой структурой* считается, что все они имеют распределенную память, а дальнейшая классификация проводится в соответствии с топологией сети: звездообразная сеть (1CAP), регулярные решетки разной размерности (Intel Paragon, CRAY T3D), гиперкубы (NCube, Intel iPCS), сети с иерархической структурой, такой, как деревья, пирамиды, кластеры (Cm, CEDAR) и, наконец, сети, изменяющие свою конфигурацию.

Заметим, что если архитектура компьютера спроектирована с использованием нескольких сетей с различной топологией, то по аналогии с гибридными MIMD с переключателями их стоит назвать *гибридными сетевыми* MIMD, а использующие идеи разных классов — просто *гибридными* MIMD. Типичным представителем последней группы, в частности, является компьютер Connection Machine 2, имеющий на внешнем уровне топологию гиперкуба, каждый узел которого является кластером.

Классификация Дж. Шора. Классификация Шора, появившаяся в начале 70-х гг., интересна тем, что представляет собой попытку выделения типичных способов компоновки вычислительных систем на основе фиксированного числа базисных блоков: устройства управления, арифметико-логического устройства, памяти команд и памяти данных. Дополнительно предполагается, что выборка из памяти данных может осуществляться словами, т. е. выбираются все разряды одного слова и/или битовым слоем — по одному разряду из одной и той же позиции каждого слова (иногда этих два способа называют горизонтальной и вертикальной выборками

соответственно). Конечно же, при анализе данной классификации надо делать скидку на время ее появления, так как предусмотреть невероятное разнообразие параллельных систем настоящего времени было в принципе невозможно. Итак, согласно классификации Шора все компьютеры разбиваются на шесть классов, которые так и называются: машина типа I, II и т. д.

Машина типа I — это вычислительная система, которая содержит устройство управления, арифметико-логическое устройство, память команд и память данных с пословной выборкой (рис. 1.12, а). Считывание данных осуществляется выборкой всех разрядов некоторого слова для их параллельной обработки в арифметико-логическом устройстве. Состав АЛУ специально не оговаривается, что допускает наличие нескольких функциональных устройств, может быть, конвейерного типа. По этим соображениям в данный класс попадают как классические последовательные машины (IBM 701, PDP-11, VAX 11/780), так конвейерные скалярные (CDC 7600) и векторно-конвейерные (CRAY-1).

Если в машине типа I осуществлять выборку не по словам, а содержимого одного разряда из всех слов, то получим машину типа II (рис. 1.12, б). Слова в памяти данных по-прежнему располагаются горизонтально, но доступ к ним осуществляется иначе. Если в машине I происходит последовательная обработка слов при параллельной обработке разрядов, то в машине II — последовательная обработка битовых слоев при параллельной обработке множества слов.

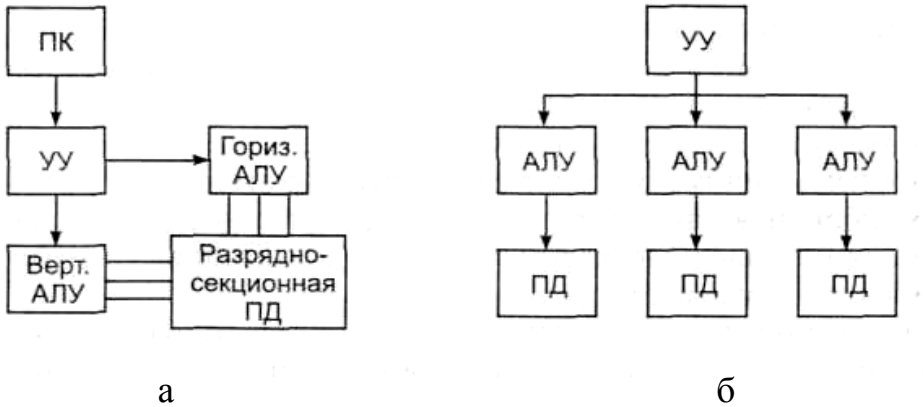
Рисунок 1.12 - Машины типов I (а) и II (б) по классификации Шора

Структура машины II лежит в основе ассоциативных компьютеров (например, центральный процессор машины STARAN), причем фактически такие компьютеры имеют не одно арифметико-логическое устройство, а множество сравнительно

простых устройств поразрядной обработки. Другим примером служит матричная система ICL DAP, которая может одновременно обрабатывать по одному разряду из 4096 слов.

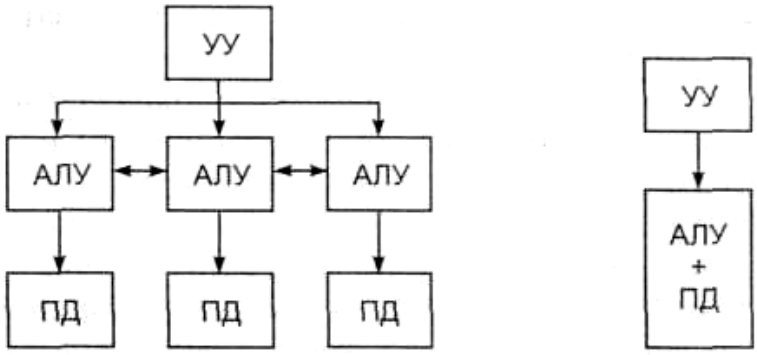
Если объединить принципы построения машин I и II, то получим машину типа III (рис. 1.13, а). Эта машина имеет два арифметико-логических устройства — горизонтальное и вертикальное, и модифицированную память данных, которая обеспечивает доступ как к словам, так и к битовым слоям. Впервые идею построения таких систем в 1960 г. выдвинул У. Шуман, называвший их ортогональными (если память представлять как матрицу слов, то доступ к данным осуществляется в направлении, «ортогональном» традиционному — не по словам (строкам), а по битовым слоям (столбцам)). В принципе, как машину STARAN, так и ICL DAP можно запрограммировать на выполнение функций машины III, но поскольку они не имеют отдельных АЛУ для обработки слов и битовых слоев, отнести их к данному классу нельзя. Полноправными представителями машин класса III являются вычислительные системы семейства OMEN-60 фирмы Sanders Associates, построенные в прямом соответствии с концепцией ортогональной машины.

Если в машине I увеличить число пар «арифметико-логическое устройство — память данных» (иногда эту пару называют процессорным элементом), то получим машину типа IV (рис. 1.13, б). Единственное устройство управления выдает команду за командой сразу всем процессорным элементам. С одной стороны, отсутствие соединений между процессорными элементами делает дальнейшее наращивание их числа относительно простым, но с другой — сильно ограничивает применимость машин этого класса. Такую структуру имеет вычислительная система PEPE, объединяющая 288 процессорных элементов.



а б

Рисунок 1.13 - Машины типов III (а) и IV (б) по классификации Шора



а б

Рисунок 1.14 - Машины типов V (а) и VI (б) по классификации Шора

Если ввести непосредственные линейные связи между соседними процессорными элементами машины IV, например, в виде матричной конфигурации, то получим схему машины типа V (рис. 1.14, а).

Любой процессорный элемент теперь может обращаться к данным как в собственной памяти, так и в памяти непосредственных соседей. Подобная структура характерна, например, для классического матричного компьютера ILLIAC IV.

Заметим, что все машины типа I—V придерживаются концепции разделения памяти данных и арифметико-логических устройств, предполагая наличие шины данных или какого-либо коммутирующего элемента между ними. Машина типа VI (рис. 1.14, б), названная матрицей с функциональной памятью (или памятью со встроенной логикой), представляет собой другой подход, предусматривающий распределение логики процессора по всему запоминающему устройству. Примерами могут служить как простые ассоциативные запоминающие устройства, так и сложные ассоциативные процессоры.

1.5.3 Современная классификация параллельных ВС

Как уже отмечалось, начало в конце 80-х годов XX в. массового выпуска микропроцессоров дало толчок бурному развитию параллельных ВС, в том числе и в области построения новых архитектур. Естественно, классификация Дункана была не в состоянии предвидеть появление новых архитектур и их систематизировать.

Построим классификацию параллельных ВС с учетом вновь появившихся современных архитектур. В основу классификации (рис. 1.15) положим четыре базовых класса Флинна (SISD, SIMD, MISD, MIMD), которые разбиваются на подклассы в соответствии с дополнениями Ванга и Бриггса.

Класс SISD разделяется на системы с одним функциональным устройством (ФУ) и несколькими ФУ.

Класс SIMD образуют два подкласса — разрядно-последовательных и пословно-последовательных ВС.

В классе MIMD выделим сильно- и слабосвязанные ВС, а также ВС, использующие идеи MIMD, подклассы которых (MIMD/SIMD, dataflow, reduction, wavefront) образуются в соответствии с классификацией Дункана.

Наконец к классу MISD отнесем ЭВМ и ВС, использующие идею конвейерной обработки. Параллелизм на уровне операторов и команд реализуется в микроконвейерных системах. При этом системы, способные разбивать на ступени непосредственное исполнение одной команды, образуют подкласс арифметико-магистральных систем, а системы, конвейеризующие все этапы выполнения команды (выборка из памяти, дешифрация, выборка операндов, исполнение, запись результатов), назовем командно-магистральными.

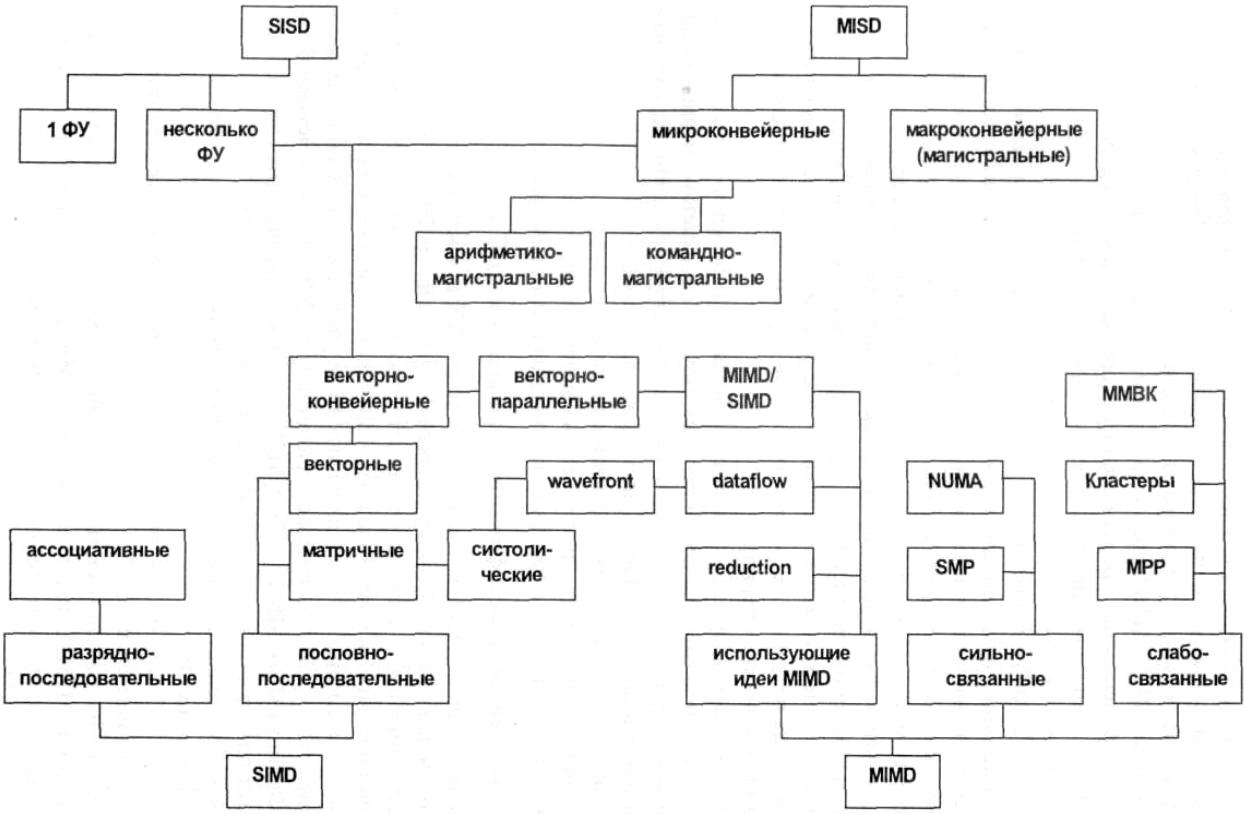


Рисунок 1.15 - Современная классификация параллельных ВС

Кроме этого, выделен подкласс макроконвейерных (магистральных) ВС, способных многократно решать одну и ту же вычислительную задачу. При этом задача разбивается на последовательные части, каждая из которых выполняется на отдельном процессоре (обрабатывающем устройстве). Выходные данные одной части являются входными для следующей. За счет организации подобного макроконвейера при многократном повторении расчетов с разными входными данными получается существенный выигрыш в производительности.

Переходя к подклассам SIMD, видно, что к разрядно-последовательным SIMD-системам можно отнести ассоциативные ВС, а к пословно-последовательным — векторные и матричные. Систолические массивы можно рассматривать как отдельный подкласс матричных архитектур.

Среди сильносвязанных MIMD-архитектур выделяют симметричные мультипроцессоры (SMP), имеющие сосредоточенную общую память и архитектуры с неоднородным доступом к памяти (NUMA), в которых логическая общая память физически распределена по узлам системы.

Слабосвязанные MIMD-системы представлены массово-параллельными (MPP), кластерными архитектурами, а также многомашинными вычислительными комплексами (ММВК). MPP-системы представляют собой совокупность *специализированных вычислительных модулей,* объединенных *высокоскоростными межпроцессорными каналами связи.*

Кластерные системы также являют собой пример массового параллелизма, только их сборка осуществляется на основе *стандартных промышленных комплектующих.* На схеме показано, что ряд ВС могут быть одновременно отнесены к нескольким классам или подклассам. Сочетание принципов микроконвейерной и векторной обработки дает векторно-конвейерные архитектуры; привнося в них идеи MIMD/SIMD, получаем векторно-параллельные (PVP) ВС. Машины wavefront являются гибридом ВС, управляемых потоком данных (dataflow) и систолических массивов.

2 Организация вычислительных систем

2.1 Многомашинные вычислительные системы

Первым типом ВС с мультиобработкой был многомашинный комплекс МК - многомашинная ВС. Здесь несколько процессоров, входящих в вычислительную систему, не имеют общей оперативной памяти, а имеют каждый свою (локальную). Каждый компьютер в многомашинной системе имеет классическую архитектуру, и такая система применяется достаточно широко. Однако эффект от применения такой вычислительной системы может быть получен только при решении задач, имеющих очень специальную структуру: она должна разбиваться на столько слабо связанных подзадач, сколько компьютеров в системе.

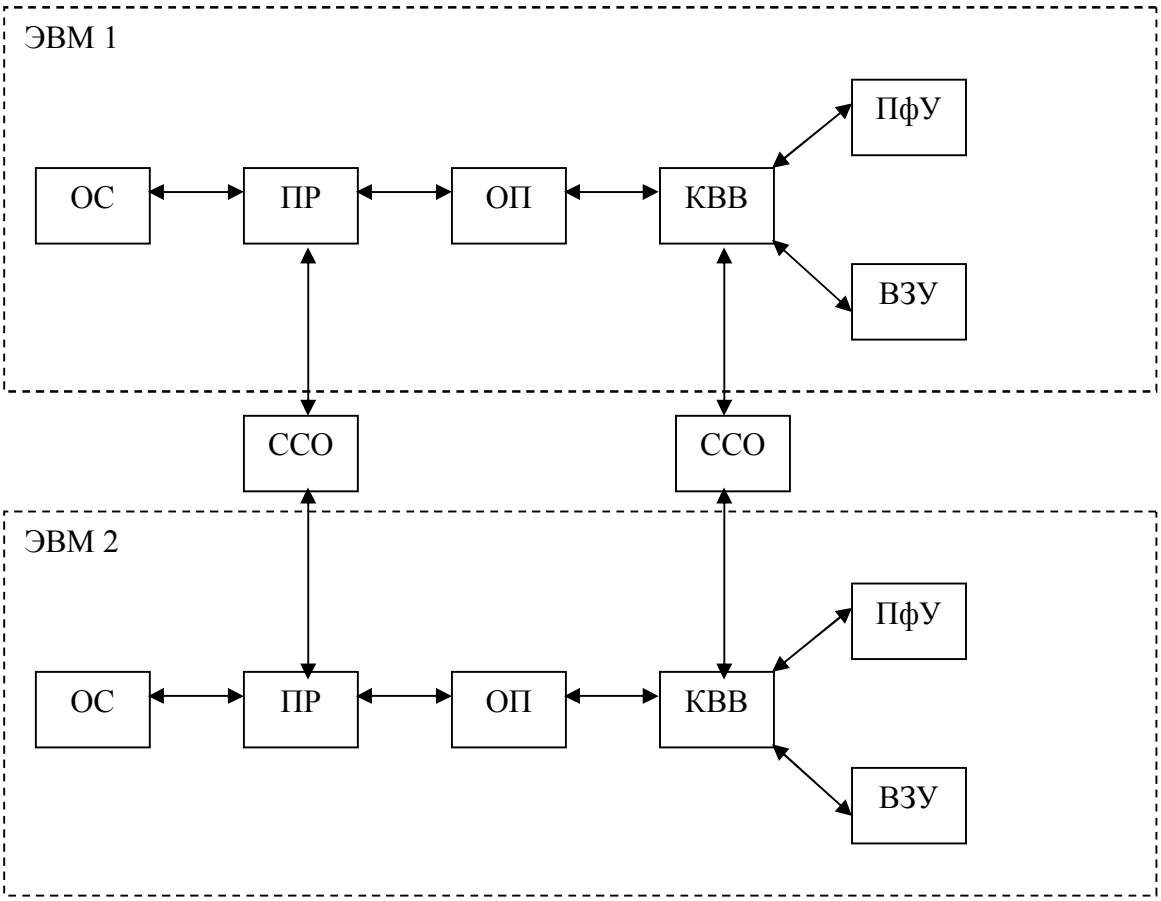


Рисунок 2.1 - Структура двухмашинной ВС

На рис. 2.1 представлена структура двухмашинной ВС. Каждая ЭВМ имеет ОП, ВЗУ, ПфУ, подключаемые к центральной части ЭВМ - процессору (ПР) с помощью каналов ввода-вывода (КВВ), и работает под управлением своей ОС. Обмен информацией между ЭВМ1 и ЭВМ2 осуществляется через системные средства обмена (ССО) в результате взаимодействия ОС машин между собой.

Основной недостаток многомашинной ВС - недостаточно эффективно используется оборудование комплекса. Достаточно, в ВС в каждой ЭВМ выйти из строя по одному устройству (даже разных типов), как вся ВС становится неработоспособной.

В настоящее время наиболее широко используют двухмашинные вычислительные комплексы, которые могут работать в одном из следующих режимов.

1. 100% - ное горячее резервирование. Обе ЭМВ в этом режиме исправны и работают параллельно, выполняя одни те же операции над одной и той же информацией (дуплексный режим). После выполнения каждой команды результаты преобразования сравниваются и при их совпадении процесс вычислений продолжается. При этом в памяти обоих ЭВМ в каждый момент находится одна и та же информация. При обнаружении несовпадения в результатах обработки Неисправная ЭВМ выводится на ремонт, а исправная ЭВМ продолжает работать под контролем встроенной в ЭВМ системы автоматического контроля.

2. 0дна исправная ЭВМ решает задачи без дублирования, а другая ЭВМ находится в режиме «Профилактика», в котором осуществляется прогон контролирующих тестов. Если основная ЭВМ продолжает не в состоянии выполнить задачу, то резервная может прекратить "Профилактику" и начать работу параллельно с основной.

3. Обе ЭВМ работают в автономном режиме со своим набором ПфУ по автономным рабочим программам.

Задание режимов работы вычислительного комплекса возможно программным путем или с помощью команд прямого управления или с пульта управления комплекса.

Наличие нескольких тесно связанных ЭВМ в составе единой ВС позволяет существенно уменьшить время вычислений благодаря параллельному выполнению на

отдельных ЭВМ различных подзадач (пакетов программ), входящих в общую задачу. 0сновное условие эффективного использования таких ВК - координация работы всех ЭВМ с помощью управляющей программы ОС, которая составляет список подзадач, подлежащих решению, и распределяет их между ЭВМ. Благодаря возможности передачи не только числовой, но и командной информации между ЭВМ можно передавать программы и части программ. Обмен программами значительно упрощает управление ВК и позволяет при загрузке какой-либо ЭВМ часть нагрузки передать другой ЭВМ, упрощает создание библиотеки стандартных программ, пригодных для любой машины ВК.

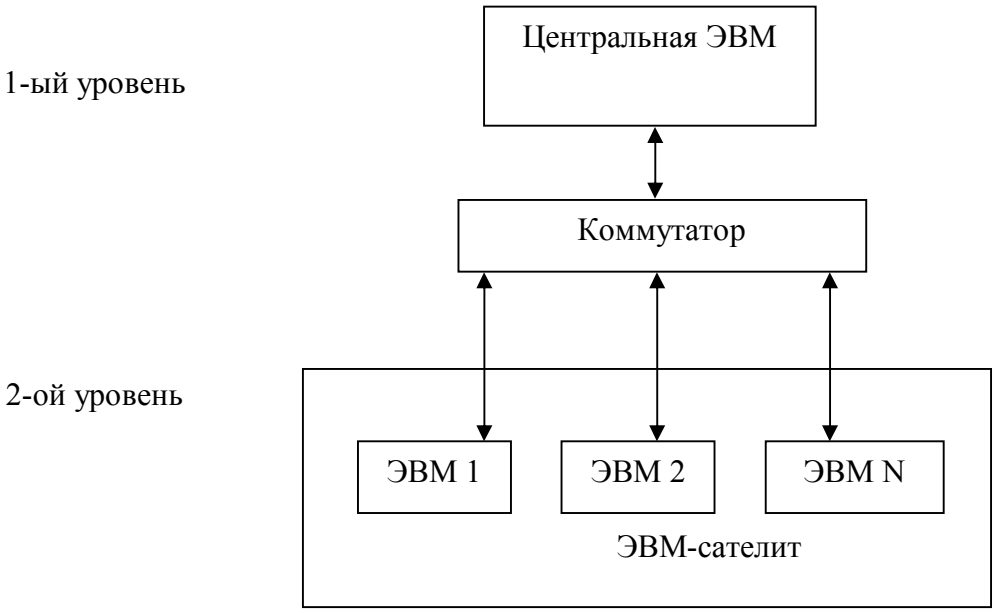


Рисунок 2.2 - Иерархическая структура многомашинной ВС

Для реализации межмашинной связи могут использоваться как средства, имеющиеся в составе ЭВМ, так и средства, предусмотренные специально для работы в составе данного ВК (см. рис 2.2).

2.2 Уровни и средства комплексирования многомашинных ВС (логические и физические уровни)

В создаваемых ВС стараются обеспечить несколько путей передачи данных, что позволяет достичь необходимой надежности функционирования, гибкости и адаптируемости к конкретным условиям работы. Эффективность обмена информацией определяется скоростью передачи и возможными

объемами данных, передаваемыми по каналу взаимодействия. Эти характеристики зависят от средств, обеспечивающих взаимодействие модулей, и уровня управления процессами, на котором это взаимодействие осуществляется.

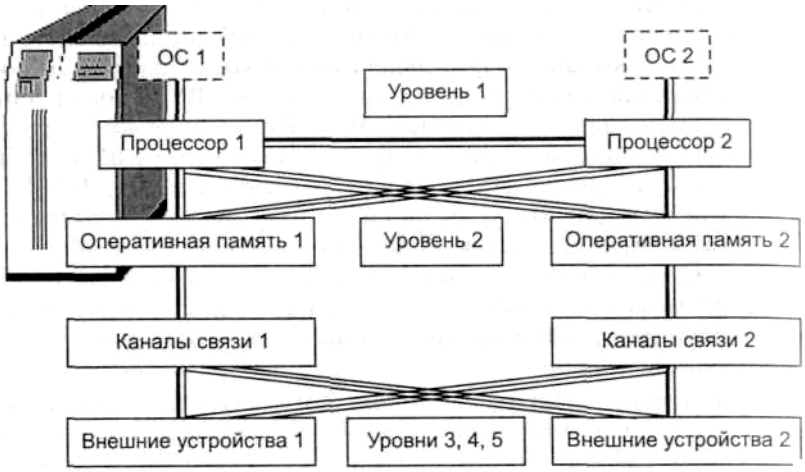


Рисунок 2.3 - Уровни комплексирования машин в вычислительную систему

Сочетание различных уровней и методов обмена данными между модулями ВС наиболее полно представлено в универсальных суперЭВМ и больших ЭВМ, в которых сбалансированно использовались основные методы достижения высокой производительности. В этих машинах предусматривались следующие уровни комплексирования (рис. 2.3):

- прямого управления (процессор — процессор);
- общей оперативной памяти;
- комплексируемых каналов ввода-вывода;
- устройств управления внешними устройствами (УВУ);
- общих внешних устройств.

На каждом из этих уровней используются специальные технические и программные средства, обеспечивающие обмен информацией.

Уровень прямого управления служит для передачи коротких однобайтовых приказов-сообщений. Последовательность взаимодействия процессоров сводится к следующему. Процессор-инициатор обмена по интерфейсу прямого управления передает в блок прямого управления байт-сообщение и подает команду «прямая запись». У другого процессора эта команда вызывает прерывание, относящееся к классу внешних. В ответ он вырабатывает команду «прямое чтение» и записывает

передаваемый байт в свою память. Затем принятая информация расшифровывается и по ней принимается решение. После завершения передачи прерывания снимаются, и оба процессора продолжают вычисления по собственным программам. Видно, что уровень прямого управления не может использоваться для передачи больших массивов данных, однако оперативное взаимодействие отдельными сигналами широко используется в управлении вычислениями. У ПЭВМ типа IBM PC этому уровню соответствует комплексирование процессоров, подключаемых к системной шине.

Уровень общей оперативной памяти (ООП) является наиболее предпочтительным для оперативного взаимодействия процессоров. В этом случае ООП эффективно работает при небольшом числе обслуживаемых абонентов.

Уровень комплектируемых каналов ввода-вывода предназначается для передачи больших объемов информации между блоками оперативной памяти, сопрягаемых в ВС. Обмен данными между ЭВМ осуществляется с помощью адаптера «канал—канал» (АКК) и команд «чтение» и «запись». Адаптер — это устройство, согласуюшее скорости работы сопрягаемых каналов. Обычно сопрягаются селекторные каналы (СК) машин как наиболее быстродействующие. Скорость обмена данными определяется скоростью самого медленного канала. Скорость передачи данных по этому уровню составляет несколько мегабайт в секунду. В ПЭВМ данному уровню взаимодействия соответствует подключение периферийной аппаратуры через контроллеры и адаптеры.

Уровень устройств управления внешними устройствами (УВУ) предполагает использование встроенного в УВУ двухканального переключателя и команд «зарезервировать» и «освободить». Двухканальный переключатель позволяет подключать УВУ одной машины к векторным каналам различных ЭВМ. По команде «зарезервировать» канат — инициатор обмена имеет доступ через УВУ к любым накопителям на дисках НМД или на магнитных лентах НМЛ. На самом деле УВУ магнитных дисков и лент — совершенно различные устройства. Обмен канала с накопителями продолжается до полного завершения работ и получения команды «освободить». Только после этого УВУ может подключиться к конкурирующему каналу. Только такая дисциплина обслуживания требований позволяет избежать конфликтных ситуаций.

На четвертом уровне *с помощью аппаратуры передачи данных(АПД)* (мультиплексоры, сетевые адаптеры, модемы и др.) имеется возможность сопряжения с каналами связи. Эта аппаратура позволяет создавать сети ЭВМ.

Пятый уровень предполагает использование *общих внешних устройств*. Для подключения отдельных устройств используется автономный двухканальный переключатель.

Пять уровней комплексирования получили название логических потому, что они объединяют на каждом уровне разнотипную аппаратуру, имеющую сходные методы управления. Каждое из устройств может иметь логическое имя, используемое в прикладных программах. Этим достигается независимость программ пользователей от конкретной физической конфигурации системы. Связь логической структуры программы и конкретной физической структуры ВС обеспечивается операционной системой по указаниям — директивам пользователя, при генерации ОС и по указаниям диспетчера-оператора вычислительного центра. Различные уровни комплексирования позволяют создавать самые различные структуры ВС.

Второй логический уровень позволяет создавать многопроцессорные ВС. Обычно он дополняется и первым уровнем, что позволяет повышать оперативность взаимодействия процессоров. Вычислительные системы сверхвысокой производительности должны строиться как многопроцессорные. Центральным блоком такой системы является быстродействующий коммутатор, обеспечивающий необходимые подключения абонентов (процессоров и каналов) к общей оперативной памяти.

Уровни 1, 3, 4, 5 обеспечивают построение разнообразных машинных комплексов. Особенно часто используется третий в комбинации с четвертым. Целесообразно их дополнять и первым уровнем.

2.3 Многопроцессорные вычислительные системы

Следующим шагом в направлении дальнейшего увеличения производительности ВС явилось создание многопроцессорных ВС с мультиобработкой. Наличие в компьютере нескольких процессоров означает, что параллельно может быть организовано много потоков данных и много потоков команд. Таким образом, параллельно могут выполняться несколько фрагментов одной задачи.

Структура такой машины, имеющей общую оперативную память и несколько процессоров, представлена на рис. 2.4. Преимущество в быстродействии многопроцессорных вычислительных систем перед однопроцессорными очевидно.

Единая ОС делает возможным автоматическое распределение ресурсов системы на различных этапах ее работы. В результате достигается высокая «живучесть» ВС, позволяющая в случае отказа отдельных модулей перераспределить нагрузку между работоспособными, обеспечив тем самым выполнение наиболее важных для ВС функций.

К недостаткам многопроцессорных ВС относят трудности, возникающие при реализации общего поля ОП, ВЗУ, а также при разработке специальной ОС. Дальнейшее развитие идей мультиобработки привело к созданию крупных многопроцессорных систем высокой производительности, получивших назначение высокопараллельных ВС. В архитектуре с параллельными процессорами, несколько АЛУ работают под управлением одного УУ. Это означает, что множество данных может обрабатываться по одной программе, т. е. по одному потоку команд. Высокое быстродействие такой архитектуры можно получить только на задачах, в которых одинаковые вычислительные операции выполняются одновременно на различных однотипных наборах данных. Структура таких ВС представлена на рис. 2.5.

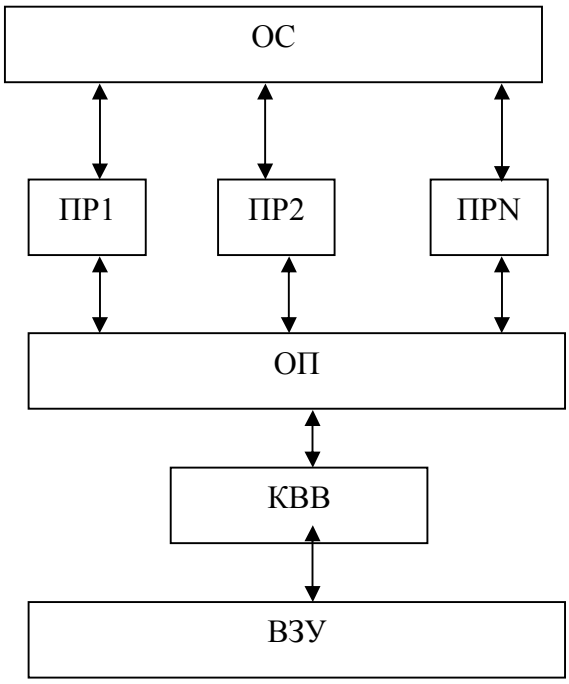


Рисунок 2.4 - Структура многопроцессорной ВС

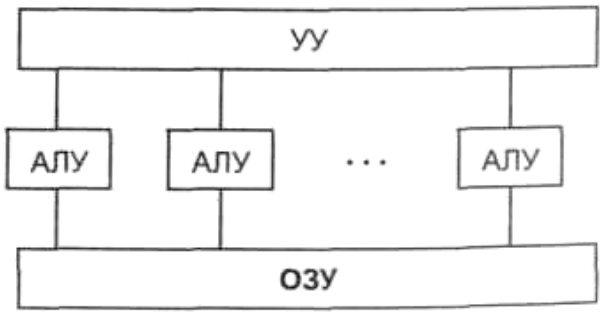


Рисунок 2.5 - Архитектура с параллельными процессорами

В настоящее время особое внимание уделяется созданию высокопараллельных ВС. Основной целью при разработке таких ВС является повышение производительности систем за счет: обеспечения возможности параллельного выполнения независимых задач; повышения эффективности работы и улучшения распределения нагрузки в системе; обеспечения наиболее экономичного обслуживания экстренных заданий и заданий при пиковых нагрузках; достижения высокого коэффициента эффективного использования ресурсов для создания новых типов архитектуры комплекса.

В высокопараллельных ВС при решении задач с небольшими емкостями памяти возможно одновременное решение на разных процессорах. Если в какой-либо интервал времени требуется резкое увеличение емкости памяти, то вся память отдается для решения задачи.

Основные особенности построения высокопараллельных ВС заключаются в следующем:

- система включает в себя один или несколько процессоров;
- центральная память системы должна находиться в общем пользовании и к ней должен быть обеспечен доступ от всех процессоров системы;
- система должна иметь общий доступ ко всем устройствам ввода-вывода, включая каналы;
- система должна иметь единую ОС, управляющую всеми аппаратными и программными средствами;
- в системе должно быть предусмотрено взаимодействие элементов аппаратного и программного обеспечения на всех уровнях: на уровне системного программного обеспечения, на программном уровне при решении задач пользователей (возможность перераспределения заданий), на уровне обмена данными и др.

Важнейшее значение для организации высокопараллельных ВС имеют способы соединения между собой различных функциональных блоков системы, так как эффективность такой системы определяется степенью параллельности или совмещения по времени работы всех устройств системы.

Чтобы дать более полное представление о многопроцессорных вычислительных системах, помимо высокой производительности необходимо назвать и другие отличительные особенности. Прежде всего это необычные архитектурные решения, направленные на повышение производительности (работа с векторными операциями организация быстрого обмена сообщениями между процессорами или организация глобальной памяти в многопроцессорных системах и др.).

2.4 Организация ВС класса SIMD

Ранее уже отмечалась нечеткость классификации Флинна, из-за чего разные типы ВС могут быть отнесены к тому или иному классу. Тем не менее, в настоящее время принято считать, что класс SIMD составляют векторные (векторно-конвейерные), матричные, ассоциативные, систолические и VLIW-вычислительные системы. Именно эти ВС будут рассмотрены ниже.

2.4.1 Векторные и векторно-конвейерные вычислительные системы

При большой размерности массивов последовательная обработка элементов матриц занимает слишком много времени, что и приводит к неэффективности универсальных ВС для рассматриваемого класса задач. Для обработки массивов требуются вычислительные средства, позволяющие с помощью единой команды производить действие сразу над всеми элементами массивов — средства векторной обработки.

В средствах векторной обработки под *вектором* понимается одномерный массив однотипных данных (обычно в форме с плавающей запятой), регулярным образом размещенных в памяти ВС. Если обработке подвергаются многомерные массивы, их также рассматривают как векторы. Такой подход допустим, если учесть, каким образом многомерные массивы хранятся в памяти ВМ. Пусть имеется массив данных *A,* представляющий собой прямоугольную матрицу размерности 4x5 (рис. 2.6).

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}

Рисунок 2.6 - Прямоугольная матрица данных

При размещении матрицы в памяти все ее элементы заносятся в ячейки с последовательными адресами, причем данные могут быть записаны строка за строкой или столбец за столбцом (рис. 2.7). С учетом такого размещения многомерных массивов в памяти вполне допустимо рассматривать их как векторы и ориентировать соответствующие вычислительные средства на обработку одномерных массивов данных (векторов).

Рисунок 2.7 - Способы размещения в памяти матрицы 4x5

Действия над многомерными массивами имеют свою специфику. Например, в двумерном массиве обработка может вестись как по строкам, так и по столбцам. Это выражается в том, с каким шагом должен меняться адрес очередного "выпираемого из памяти элемента. Так, если рассмотренная в примере матрица расположена в памяти построчно, адреса последовательных элементов строки различаются на единицу, а для элементов столбца шаг равен пяти. При размещении матрицы по столбцам единице будет равен шаг по столбцу, а шаг по строке — четырем. В векторной концепции для

обозначения шага, с которым элементы вектора извлекаются из памяти, применяют термин *шаг по индексу (stride)*.

Еще одной характеристикой вектора является число составляющих его элементов — *длина вектора*.

Векторный процессор — это процессор, в котором операндами некоторых команд могут выступать упорядоченные массивы данных - векторы. Векторный процессор может быть реализован в двух вариантах. В первом он представляет собой дополнительный блок к универсальной вычислительной машине (системе). Во втором — векторный процессор - это основа самостоятельной ВС.

Рассмотрим возможные подходы к архитектуре средств векторной обработки. Наиболее распространенные из них сводятся к трем группам:

- конвейерное АЛУ;
- массив АЛУ;
- массив процессорных элементов.

Последний вариант — один из случаев многопроцессорной системы, известной как *матричная ВС*. Понятие векторного процессора имеет отношение к двум первым группам, причем, как правило, к первой (см. рис. 2.8).

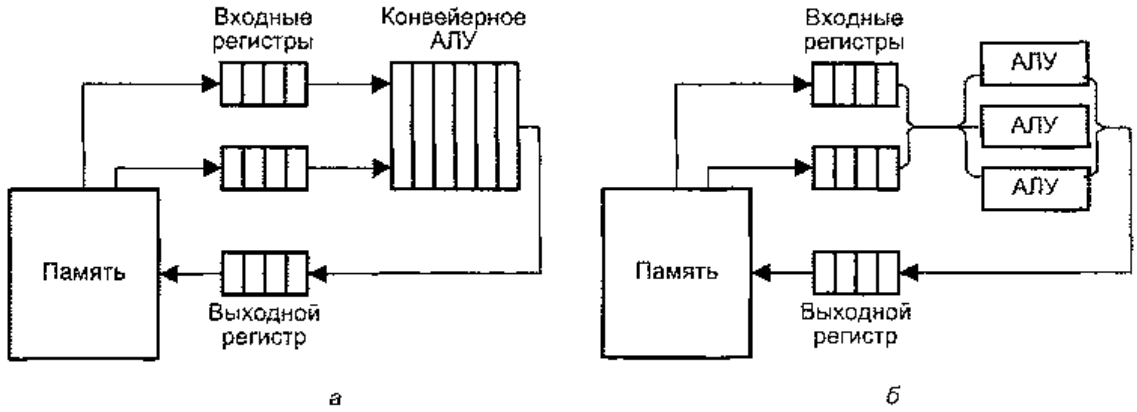


Рисунок 2.8 - Варианты векторных вычислений:
а) с конвейерным АЛУ; б) с несколькими АЛУ

В варианте с конвейерным АЛУ (рис. 2.8, а) обработка элементов векторов производится конвейерным АЛУ для чисел с плавающей запятой (ПЗ). Операции с числами в форме с ПЗ достаточно сложны, но поддаются разбиению на отдельные шаги. Так, сложение двух чисел может быть сведено к четырем этапам: сравнению

порядков, сдвигу мантиссы меньшего из чисел, сложению мантисс и нормализации результата (рис. 2.9, а).

Каждый этап может быть реализован с помощью отдельной ступени конвейерного АЛУ (рис. 2.9, б). Очередной элемент вектора подается на вход конвейера, как только освобождается первая ступень (рис. 2.9, в). Ясно, что такой вариант вполне годится для обработки векторов.

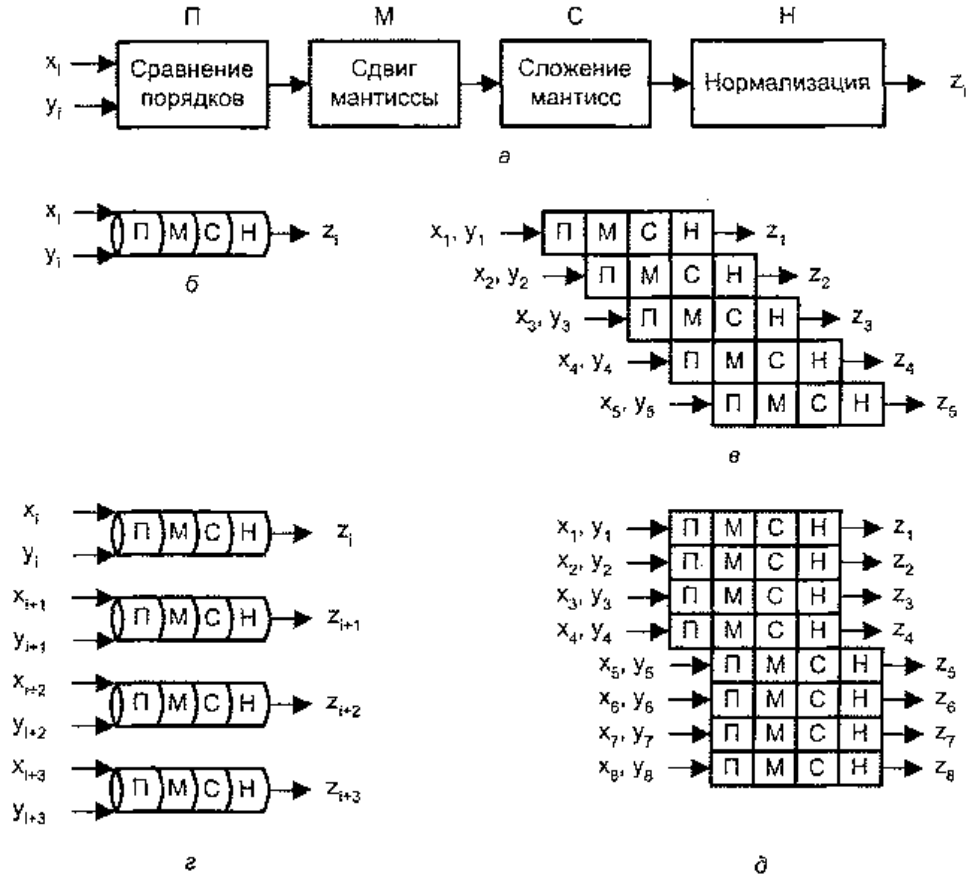


Рисунок 2.9 - Обработка векторов: а) структура арифметического конвейера для чисел с плавающей запятой; б) обозначение конвейера; в) обработка векторов конвейерными АЛУ; д) конвейерная обработка векторов четырьмя АЛУ

Одновременные операции над элементами векторов можно производить с помощью нескольких параллельно используемых АЛУ, каждое из которых отвечает за одну пару элементов (см. рис. 2.8, б). Такого рода обработка, когда каждое из АЛУ является конвейерным, показана на рис. 2.9, г.

Если параллельно используются конвейерные АЛУ, то возможен еще один уровень конвейеризации, что иллюстрирует рис. 2.9, д. Вычислительные системы, где

реализована эта идея, называют *векторно-конвейерными*. Коммерческие векторно-конвейерные ВС, в состав которых для обеспечения универсальности включен также скалярный процессор, известны как *суперЭВМ*.

Обобщенная структура векторного процессора приведена па рис. 2.10. На схеме показаны основные узлы процессора, без детализации некоторых связей между ними.

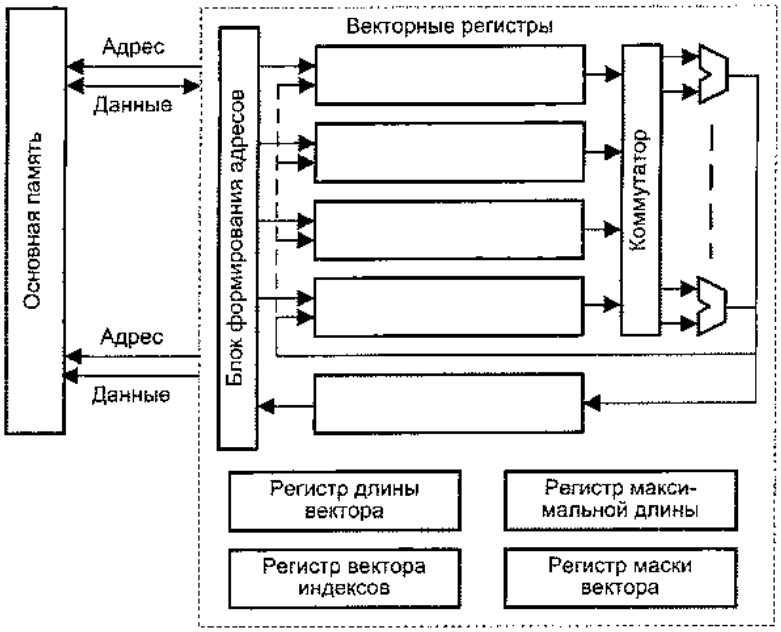


Рисунок 2.10 - Упрощенная структура векторного процессора

Обработка всех *n* компонентов векторов-операндов задается одной векторной командой. Общепринято, что элементы векторов представляются числами в форме с плавающей запятой (ПЗ). АЛУ векторного процессора может быть реализовано в виде единого конвейерного устройства, способного выполнять все предусмотренные операции над числами с ПЗ. Более распространена, однако, иная структура, - в ней АЛУ состоит из отдельных блоков сложения и умножения, а иногда и блока для вычисления обратной величины, когда операция деления *X/Y* реализуется в виде *X(1/Y)*. Каждый из таких блоков также конвейеризирован.

Кроме того, в состав векторной вычислительной системы обычно включают и скалярный процессор, что позволяет параллельно выполнять векторные и скалярные команды.

Для хранения векторов-операндов вместо множества скалярных регистров используют так называемые векторные регистры, которые представляют собой совокупность скалярных регистров, объединенных в очередь типа *FIFO*, способную

хранить 50-100 чисел с плавающей запятой. Набор векторных регистров (*Va, Vb, Vc*,...) имеется в любом векторным процессоре. Система команд векторного процессора поддерживает работу с векторными регистрами и обязательно включает и себя команды:

- загрузки векторного регистра содержимым последовательных ячеек памяти, указанных адресом первой ячейки этой последовательности;
- выполнения операций над всеми элементами векторов, находящихся в векторных регистрах;
- сохранения содержимого векторного регистра в последовательности ячеек памяти, указанных адресом первой ячейки этой последовательности.

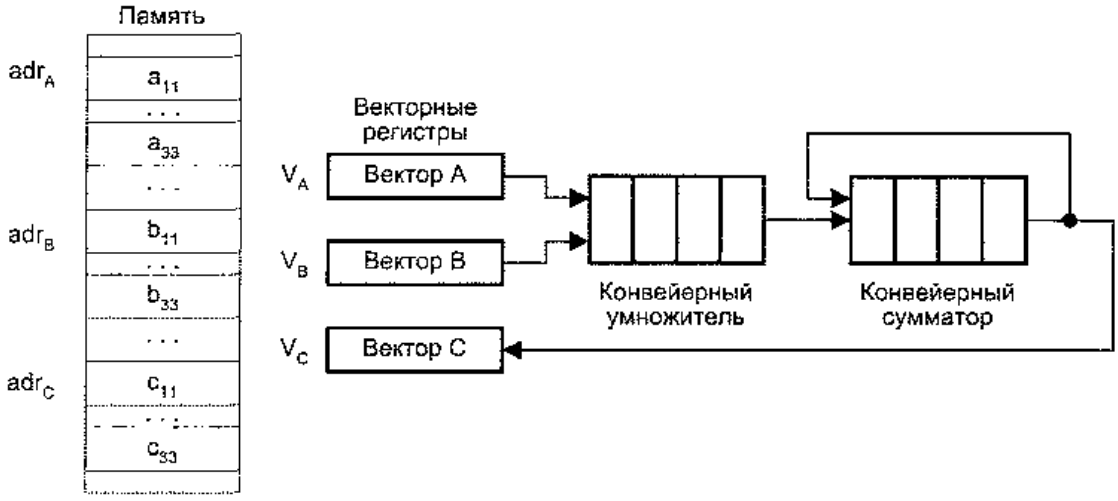


Рисунок 2.11 - Векторный процессор для вычисления скалярного произведения

Векторный процессор с конвейеризированными блоками обработки для вычисления скалярного произведения показан на рис. 2.11.

Векторы A и B, хранящиеся в памяти начиная с адресов adr_A и adr_B, загружаются в векторные регистры V_A и V_B соответственно. Предполагается, что конвейерные умножитель и сумматор состоят из четырех сегментов, которые вначале инициализируются нулем, поэтому в течение первых восьми циклов, пока оба конвейера не заполнятся, на выходе сумматора будет 0. Пары (A_i, B_i) подаются на вход умножителя и перемножаются в темпе одна пара за цикл. После первых четырех циклов произведения начинают суммироваться с данными, поступающими с выхода сумматора. В течение следующих четырех циклов на вход сумматора поступают суммы произведений из умножителя с нулем. К концу восьмого цикла в сегментах

сумматора находятся четыре первых произведения A_1B_1, ..., A_4B_4, а в сегментах умножителя — следующие четыре произведения: A_5B_5, …, A_8B_8. К началу девятого цикла на выходе сумматора будет A_1B_1, а на выходе умножителя - A_5B_5. Таким образом, десятый цикл начнется со сложения в сумматоре A_2B_2 и A_5B_5. Десятый цикл начнется со сложения $A_2B_2 + A_6B_6$ и т. д. Процесс суммирования в четырех секциях выглядит так:

$$\begin{aligned}\mathbf{C} = {} & \mathbf{A_1B_1} + \mathbf{A_5B_5} + \mathbf{A_9B_9} + \mathbf{A_{13}B_{13}} + \ldots \\ & + \mathbf{A_2B_2} + \mathbf{A_6B_6} + \mathbf{A_{10}B_{10}} + \mathbf{A_{14}B_{14}} + \ldots \\ & + \mathbf{A_3B_3} + \mathbf{A_7B_7} + \mathbf{A_{11}B_{11}} + \mathbf{A_{15}B_{15}} + \ldots \\ & + \mathbf{A_4B_4} + \mathbf{A_8B_8} + \mathbf{A_{12}B_{12}} + \mathbf{A_{16}B_{16}} + \ldots\end{aligned}$$

Когда больше не остается членов для сложения, система заносит в умножитель четыре нуля. Конвейер сумматора в своих четырех сегментах при этом будет содержать четыре скалярных произведения, соответствующие четырем суммам, приведенным в четырех строках показанного выше уравнения. Далее четыре частичных суммы складываются для получения окончательного результата.

Программа для вычисления скалярного произведения векторов A и B, хранящихся в двух областях памяти с начальными адресами adr_A и adr_B, соответственно может выглядеть так:

```
V_load V_A, adr_A
V_load V_B, adr_B
V_multiply V_C, V_A, V_B.
```

Первые две векторные команды V_load загружают векторы из памяти в векторные регистры V_A и V_B. Векторная команда умножения V_multiply вычисляет произведение для всех пар одноименных элементов векторов и записывает полученный вектор в векторный регистр V_C.

Важным элементом любого векторного процессора (ВП) является *регистр длины вектора*. Этот регистр определяет, сколько элементов фактически содержит обрабатываемый в данный момент вектор, то есть сколько индивидуальных операций с элементами нужно сделать. В некоторых ВП присутствует также регистр максимальной длины вектора, определяющий максимальное число элементов вектора, которое может быть одновременно обработано аппаратурой процессора. Этот регистр используется при разделении очень длинных векторов на сегменты, длина которых

соответствует максимальному числу элементов, обрабатываемых аппаратурой за один прием.

Достаточно часто приходится выполнять такие операции, в которых должны участвовать не все элементы векторов. Векторный процессор обеспечивает данный режим с помощью *регистра маски вектора*. В этом регистре каждому элементу вектора соответствует один бит. Установка бита в единицу разрешает запись соответствующего элемента вектора результата в выходной векторный регистр, а сброс в ноль — запрещает.

Как уже упоминалось, элементы векторов в памяти расположены регулярно, и при выполнении векторных операций достаточно указать значение шага по индексу. Существуют, однако, случаи, когда необходимо обрабатывать только ненулевые элементы векторов. Для поддержки подобных операций в системе команд ВП предусмотрены операции *упаковки/распаковки* (gather/scatter). Операция упаковки формирует вектор, содержащий только ненулевые элементы исходного вектора, а операция распаковки производит обратное преобразование. Обе этих задачи векторный процессор решает с помощью вектора индексов, для хранения которого используется регистр вектора индексов, по структуре аналогичный регистру маски. В векторе индексов каждому элементу исходного вектора соответствует один бит. Нулевое значение бита свидетельствует, что соответствующий элемент исходного вектора равен нулю.

Использование векторных команд окупается благодаря двум качествам. Во-первых, вместо многократной выборки одних и тех же команд достаточно произвести выборку только одной векторной команды, что позволяет сократить издержки за счет устройства управления и уменьшить требования к пропускной способности памяти. Во-вторых, векторная команда обеспечивает процессор упорядоченными данными. Когда инициируется векторная команда, ВС знает, что ей нужно извлечь n пар операндов, расположенных в памяти регулярным образом. Таким образом, процессор может указать памяти на необходимость начать извлечение таких пар. Если используется память с чередованием адресов, эти пары могут быть получены со скоростью одной пары за цикл процессора и направлены для обработки в конвейеризированный функциональный блок. При отсутствии чередования адресов

или других средств извлечения операндов с высокой скоростью преимущества обработки векторов существенно снижаются.

2.4.2 Матричные вычислительные системы

Назначение матричных вычислительных систем во многом схоже с назначением векторных ВС — обработка больших массивов данных. В основе матричных систем процессорных элементов (ПЭ). Организация систем подобного типа на первый взгляд достаточно проста. Они имеют общее управляющее устройство, генерирующее поток команд, и большое число ПЭ, работающих параллельно и обрабатывающих каждый свой ноток данных. Однако па практике, чтобы обеспечить достаточную эффективность системы при решении широкого круга задач, необходимо организовать связи между процессорными элементами так, чтобы наиболее полно загрузить процессоры работой. Именно характер связей между ПЭ и определяет разные свойства системы. Ранее уже отмечалось, что подобная схема применима и для векторных вычислении.

Между матричными и векторными системами есть существенная разница. Матричный процессор интегрирует множество идентичных *функциональных блоков* (ФБ), логически объединенных в матрицу и работающих в SIMD-стиле. Не столь существенно, как конструктивно реализована матрица процессорных элементов — на едином кристалле пли на нескольких. Важен сам принцип — ФБ логически скомпонованы в матрицу и работают синхронно, то есть присутствует только один поток команд для всех. Векторный процессор имеет встроенные команды для обработки векторов данных, что позволяет эффективно загрузить конвейер из функциональных блоков. В свою очередь, векторные процессоры проще испол1.::овать, потому что команды для обработки векторов — это более удобная для человека модель программирования, чем SIMD.

Структуру матричной вычислительной системы можно представить в виде, показанном на рис. 2.12.

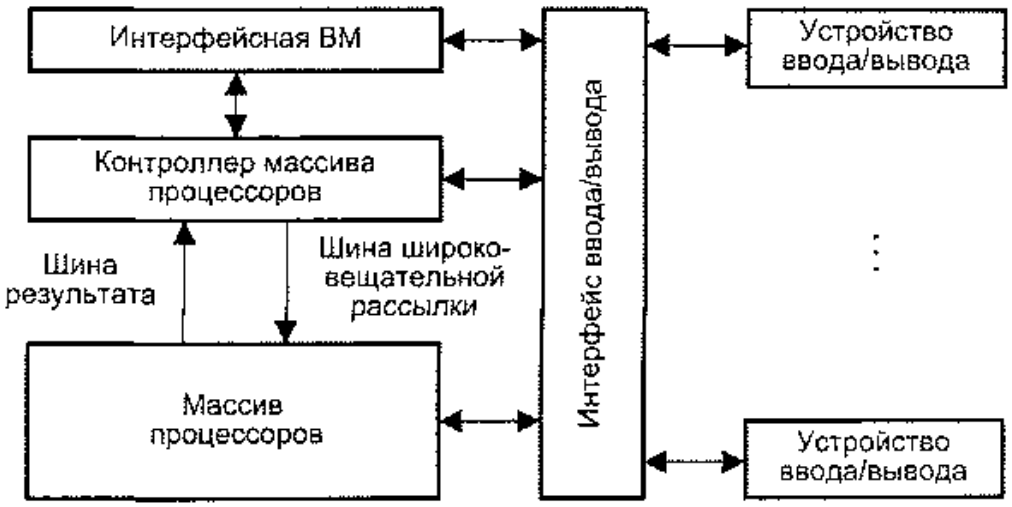


Рисунок 2.12 - Обобщенная модель матричной SIMD-системы

Собственно параллельная обработка множественных элементов данных осуществляется массивом процессоров (МПр). Единый поток команд, управляющий обработкой данных в массиве процессоров, генерируется *контроллером массива процессоров* (КМП).

КМП выполняет последовательный программный код, реализует операции условного и безусловного переходов, транслирует в МПр команды, данные и сигналы управления. Команды обрабатываются процессорами в режиме жесткой синхронизации. Сигналы управления используются для синхронизации команд и пересылок, а также для управления процессом вычислений, в частности определяют, какие процессоры массива должны выполнять операцию, а какие — нет. Команды, данные и сигналы управления передаются из КМП в массив процессоров но шине *широковещательной рассылки*. Поскольку выполнение операций условного перехода зависит от результатов вычислений, результаты обработки данных в массиве процессоров транслируются в КМП, проходя по шине результата.

Для обеспечения пользователя удобным интерфейсом при создании и отладке программ в состав подобных ВС обычно включают *интерфейсную ВМ* (ИВМ, front-end computer). В роли такой ВМ выступает универсальная вычислительная машина, на которую дополнительно возлагается задача загрузки программ и данных в КМП. Кроме того, загрузка программ и данных в КМП может производиться и напрямую с *устройств ввода/вывода*, например с магнитных дисков. После загрузки КМП

приступает к выполнению программы, транслируя в МПр по широковещательной шине соответствующие SIMD-команды.

Рассматривая массив процессоров, следует учитывать, что для хранения множественных наборов данных в нем, помимо множества процессоров, должно присутствовать сеть взаимосвязей, как между процессорами, так и между процессорами и модулями памяти. Таким образом, под термином массив процессоров понимают блок, состоящий из процессоров, модулей памяти и сети соединений.

Дополнительную гибкость при работе с рассматриваемой системой обеспечивает механизм маскирования, позволяющий привлекать к участию в операциях лишь определенное подмножество из входящих в массив процессоров. Маскирование реализуется как на стадии компиляции, так и на этапе выполнения, при этом процессоры, исключенные путем установки в ноль соответствующих битов маски, во время выполнения команды простаивают.

Контроллер массива процессоров выполняет последовательный программный код, реализует команды ветвления программы, транслирует команды и сигналы управления в процессорные элементы. Рисунок 2.13 иллюстрирует одну из возможных реализаций КМП.

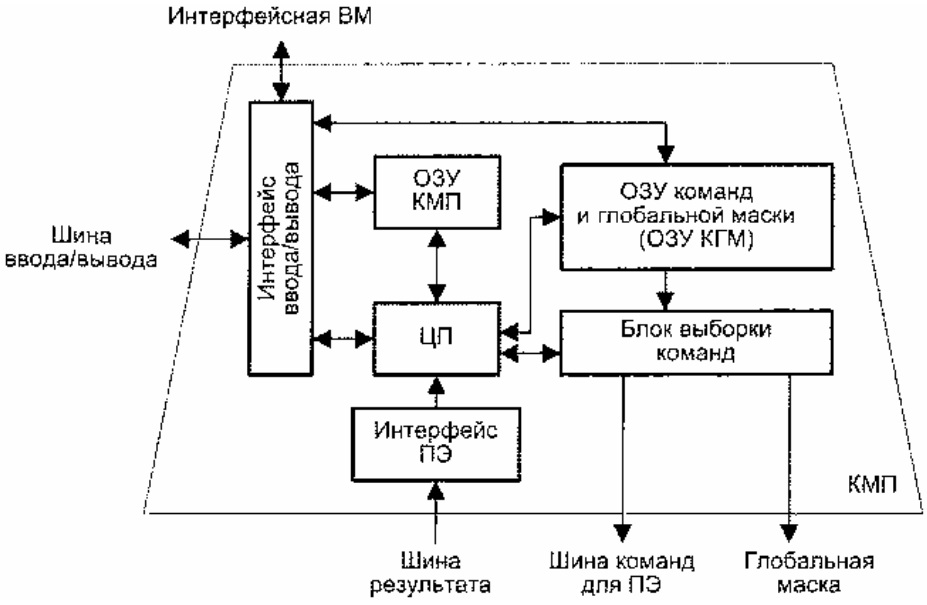
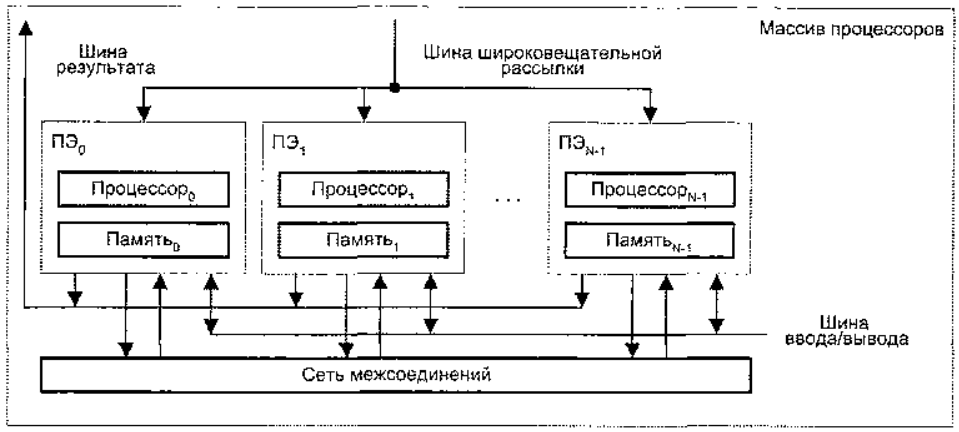


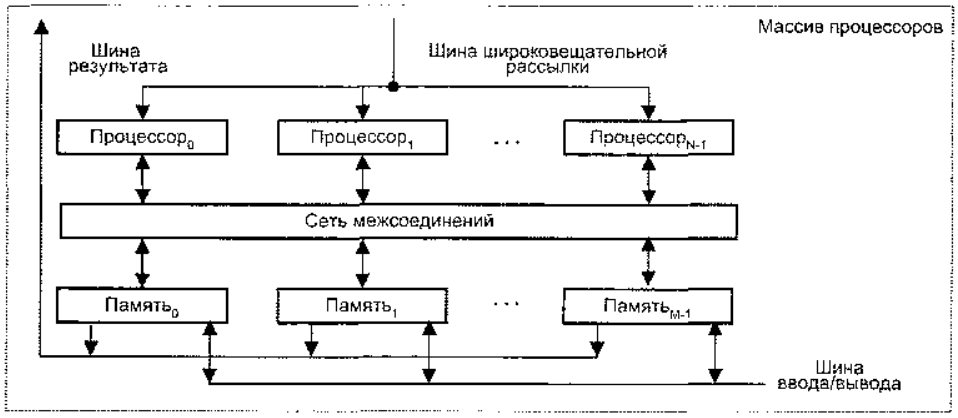
Рисунок 2.13 - Модель контроллера массива процессоров

При загрузке из ИВМ программа через интерфейс ввода/вывода заносится в оперативное запоминающее устройство КМП (ОЗУ КМП). Команды для процессорных элементов и глобальная маска, формируемая на этапе компиляции, так-

же через интерфейс ввода/вывода загружаются в ОЗУ команд и глобальной маски (ОЗУ КГМ). Затем КМП начинает выполнять программу, извлекая либо одну скалярную команду из ОЗУ КМП, либо множественные команды из ОЗУ КГМ. Скалярные команды — команды, осуществляющие операции нал хранящимися в КМП скалярными данными, выполняются центральным процессором (ЦП) контроллера массива процессоров. В свою очередь, команды, оперирующие параллельными переменными, хранящимися в каждом ПЭ, преобразуются в блоке выборки команд в более простые единицы выполнения - *нанокоманды*. Нанокоманды совместно с маской пересылаются через шину команд для ПЭ ид исполнение в массив процессоров. Например, команда сложения 32-разрядпых слов в КМП системы МРР преобразуется в 32 нанокоманды одноразрядного сложения, которые каждым ПЭ обрабатываются последовательно.



а



б

Рисунок 2.14 – Модель массивов процессоров: а) «процессорный элемент-процессорный элемент, б) «процессор-память»

В большинстве алгоритмов дальнейший порядок вычислений зависит от результатов и/или флагов условий предшествующих операций. Для обеспечения такого режима в матричных системах статусная информация, хранящаяся и процессорных элементах, должна быть собрана в единое слово и передана в КМП для выработки решения о ветвлении программы.

В матричных SIMD-системах распространение получили два основных типа архитектурной организации массива процессорных элементов (рис. 2.14).

В первом варианте, известном как архитектура типа «процессорный элемент-процессорный элемент» («ПЭ-ПЭ»), N процессорных элементов (ПЭ) связаны между собой сетью соединении (рис. 2.14, а). Каждый ПЭ — это процессор с локальной памятью. Процессорные элементы выполняют команды, получаемые из КМП по шине широковещательной рассылки, и обрабатывают данные как хранящиеся в их локальной памяти, так и поступающие из КМП. Обмен данными между процессорными элементами производиться по сети соединений, в то время как шина ввода/вывода служит для обмена информацией между ПЭ и устройствами ввода/вывода. Для трансляции результатов из отдельных ПЭ в контроллер массива процессоров служит шина результата. Благодаря использованию локальной памяти аппаратные средства ВС рассматриваемого типа могут быть построены весьма эффективно. Во многих алгоритмах действия по пересылке информации по большей части локальны, то есть происходят между ближайшими соседями. По этой причине архитектура, где каждый ПЭ связан только с соседними, очень популярна. В качестве примеров вычислительных систем с рассматриваемой архитектурой можно упомянуть MasPar MP-1, Connection Machine CM-2, GF11, DAP, MPP, STARAN, PEPE, ILLIAC IV.

Второй вид архитектуры - «процессор-память» — показан на рис. 2.14,б. В такой конфигурации двунаправленная сеть соединений связывает N процессоров с M модулями памяти. Процессоры управляются КМП через широковещательную шину. Обмен данными между процессорами осуществляется как через сеть, так и через модули памяти. Пересылка данных между модулями памяти и устройствами ввода/вывода обеспечивается шиной ввода/вывода. Для передачи данных из конкретного модуля памяти в КМП служит шина результата. Примерами ВС с

рассмотренной архитектурой могут служить Burroughs Scientific Processor (BSP), Texas Reconfigurable Array Computer TRAC.

В большинстве матричных SIMD-систем в качестве процессорных элементов применяются простые RISC-процессоры с локальной памятью ограниченной емкости.

Неотъемлемыми компонентами ПЭ (рис. 2.15) в большинстве вычислительных систем являются:

- арифметико-логическое устройство (АЛУ);
- регистры данных;
- сетевой интерфейс (СИ), который может включать в свой состав регистры пересылки данных;
- номер процессора;
- регистр флага разрешения маскирования (F);
- локальная память.

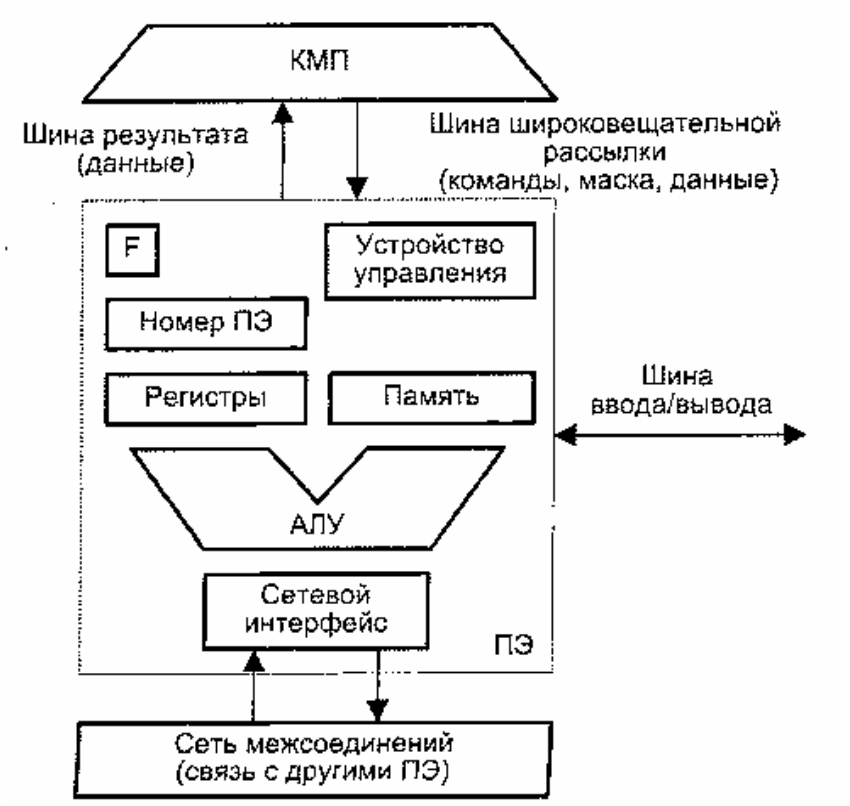


Рисунок 2.15 – Модель процессорного элемента

Процессорные элементы, управляемые командами, поступающими по широковещательной шине из КМП, могут выбирать данные из своей локальной памяти и регистров, обрабатывать их в АЛУ и сохранять результаты в регистрах и локальной памяти. ПЭ могут также обрабатывать те данные, которые поступают по

нише широковещательной рассылки из КМП. Кроме того, каждый процессорный элемент вправе получать данные из других ПЭ и отправлять их в другие ПЭ по сети соединений, используя для этого свой сетевой интерфейс. В некоторых матричных системах, в частности в MasPar MP-1, элемент данных из ПЭ-источника можно передавать в ПЭ-приемник непосредственно, в то время как в других, например в MPP, — данные предварительно должны быть помешены в специальный регистр пересылки данных, входящий в состав сетевого интерфейса. Пересылка данных между ПЭ и устройствами ввода/вывода осуществляется через шину ввода/вывода ВС. В ряде систем (MasPar MP-1) ПЭ подключены к шине ввода/вывода посредством сети соединений и канала ввода/вывода системы. Результаты вычислений любое ПЭ выдает в КМП через шину результата.

Каждому из N ПЭ в массиве процессоров присваивается уникальный номер, называемый также адресом ПЭ, который представляет собой целое число от 0 до .V-1. Чтобы указать, должен ли данный ПЭ участвовать в общей операции, в его составе имеется регистр флага разрешения F. Состояние этого регистра определяют сигналы управления из КМП, либо результаты операций в самом ПЭ, либо и те и другие совместно.

Еще одной существенной характеристикой матричной системы является способ синхронизации работы ПЭ. Так как все ПЭ получают и выполняют команды одновременно, их работа жестко синхронизируется. Это особенно важно в операциях пересылки информации между ПЭ. В системах, где обмен производится с четырьмя соседними ПЭ, передача информации осуществляется в режиме «регистр-регистр».

2.4.3 Ассоциативные вычислительные системы

К классу SIMD относятся и так называемые ассоциативные вычислительные системы. В основе подобной ВС лежит ассоциативное запоминающее устройство, а точнее - ассоциативный процессор, построенный на базе такого ЗУ. Напомним, что ассоциативная намять (или ассоциативная матрица) представляет собой ЗУ, где выборка информации осуществляется не по адресу операнда, а по отличительным признакам операнда. Запись в традиционное ассоциативное ЗУ также производится не по адресу, а в одну из незанятых ячеек.

Ассоциативный процессор (АП) можно определить как ассоциативную память, допускающую параллельную запись во все ячейки, для которых было зафиксировано

совпадение с ассоциативным признаком. Эта особенность АП, носящая название *мультизаписи*, является первым отличием ассоциативного процессора от традиционной ассоциативной памяти. Считывание и запись информации могут производиться по двум срезам запоминающего массива - либо это все разряды одного слова, либо один и тот же разряд всех слов. При необходимости выделения отдельных разрядов среза лишние позиции допустимо маскировать. Каждый разряд среза в АП снабжен собственным процессорным элементом, что позволяет между считыванием информации и ее записью производить необходимую обработку, то есть параллельно выполнять операции арифметического сложения, поиска, а также эмулировать многие черты матричных ВС, таких, например, как ILLIAC IV.

Таким образом, ассоциативные ВС или ВС с ассоциативным процессором есть те что иное, как одна из разновидностей параллельных ВС, в которых *n* процессорных элементов ПЭ (вертикальный разрядный срез памяти) представляют собой простые устройства, как правило, последовательной поразрядной обработки. При этом каждое слово (ячейка) ассоциативной памяти имеет свое собственное устройство обработки данных (сумматор). Операция осуществляется одновременно всеми *n* IIЭ. Все или часть элементарных последовательных ПЭ могут синхронно выполнять операции над всеми ячейками или над выбранным множеством слов ассоциативной памяти.

Время обработки *N* *m*-разрядных слов в ассоциативной ВС определяется выражением:

$$T = m * t * \left(\frac{N}{n} + K \right)$$

где t — время цикла ассоциативной памяти; *n* — число ячеек ассоциативной системы; К— коэффициент сложности выполнения элементарной операции (количество последовательных шагов, каждый из которых связан с доступом к памяти).

2.4.4 Вычислительные системы с систолической структурой

В фон-неймановских машинах данные, считанные из памяти, однократно обрабатываются в процессорном элементе, после чего снопа возвращаются в память (рис. 2.16, а). Авторы идеи систолической матрицы Кунг и Лейзерсон предложили организовать вычисления так, чтобы данные на своем пути от считывания из памяти

до возвращения обратно пропускались через как можно большее число ПЭ (рис. 2.16, б).

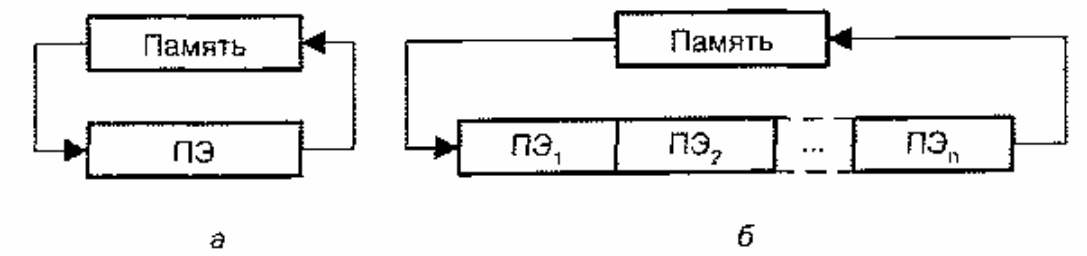


Рисунок 2.16 - Обработка данных в ВС:
а) фон-неймановского типа; б) систолической структуры

Если сравнить положение памяти в ВС со структурой живого организма, то по аналогии ей можно отвести роль сердца, множеству ПЭ — роль тканей, а поток данных рассматривать как циркулирующую кровь. Отсюда и происходит название *систолическая матрица* (*систола — сокращение предсердий и желудочков сердца при котором кровь нагнетается в артерии*). Систолические структуры эффективны при выполнении матричных вычислений, обработке сигналов, сортировке данных и т. д. В качестве примера авторами идеи был предложен линейный массив для алгоритма матричного умножения, показанный на рис. 2.17.

В основе схемы лежит ритмическое прохождение двух потоков данных x_i, и y_i навстречу друг другу. Последовательные элементы каждого потока разделены одним тактовым периодом, чтобы любой из них мог встретиться с любым элементом встречного потока. Если бы они следовали в каждом периоде, то элемента x_i, никогда бы не встретился с элементами y_{i+1} , y_{i+3},... .

Вычисления выполняются параллельно в процессорных элементах, каждый из которых реализует один шаг в операции вычисления скалярного произведения (IPS, Inner Product Step) и носит название *IPS-элемента* (рис. 2.18).

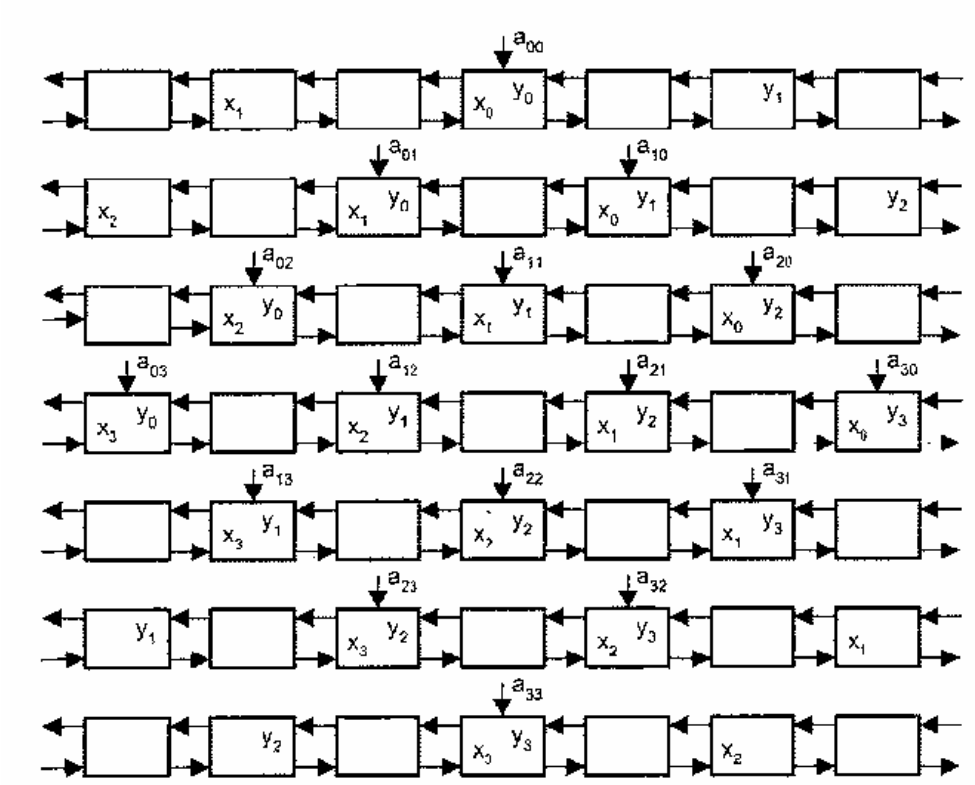


Рисунок 2.17 - Процесс векторного умножения матриц (*n = 4*)

Значение $y_{вх}$, поступающее на вход ПЭ, суммируется с произведением входных значений $x_{вх}$ и $a_{вх}$. Результат выходит из ПЭ как $y_{вых}$. Значение $x_{вх}$, кроме того, для возможного последующего использования остальной частью массива транслируется через ПЭ без изменений и покидает его в виде $x_{вых}$.

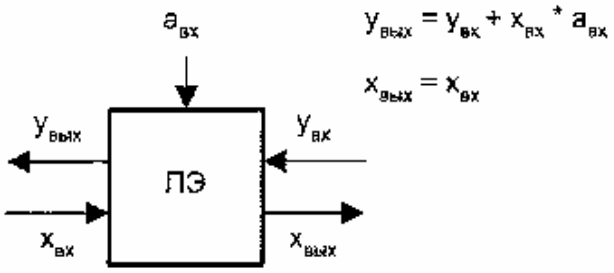


Рисунок 2.18 - Функциональная схема IPS-элемента

Таким образом, *систолическая структура* – это однородная вычислительная среда из процессорных элементов, совмещающая в себе свойства конвейерной и матричной обработки и обладающая следующими особенностями:

вычислительный процесс в систолических структурах представляет собой непрерывную и регулярную передачу данных от одного ПЭ к другому без запоминания промежуточных результатов вычисления;

- каждый элемент входных данных выбирается из памяти однократно и используется столько раз, сколько необходимо по алгоритму, ввод данных осуществляется в крайние ПЭ матрицы;
- образующие систолическую структуру ПЭ однотипны и каждый из них может быть менее универсальным, чем процессоры обычных многопроцессорных систем;
- потоки данных и управляющих сигналов обладают регулярностью, что позволяет объединять ПЭ локальными связями минимальной длины;
- алгоритмы функционирования позволяют совместить параллелизм с конвейерной обработкой данных;
- производительность матрицы можно улучшить за счет добавления в нее определенного числа ПЭ, причем коэффициент повышения производительности при этом линеен.

В настоящее время достигнута производительность систолических процессоров порядка 1000 млрд операций/с.

На данный момент разработаны систолические матрицы с различной геометрией связей: линейные, квадратные, гексагональные, трехмерные и др. Перечисленные конфигурации систолических матриц приведены на рис. 2.19.

Каждая конфигурация матрицы наиболее приспособлена для выполнения определенных функций, например линейная матрица оптимальна для реализации фильтров в реальном масштабе времени; гексагональная — для выполнения операций обращения матриц, а также действий над матрицами специального вила (Теплица-Генкеля); трехмерная - для нахождения значений нелинейных дифференциальных уравнений в частных производных или для обработки сигналов антенной решетки. Наиболее универсальными и наиболее распространенными, тем не менее, можно считать матрицы с линейной структурой.

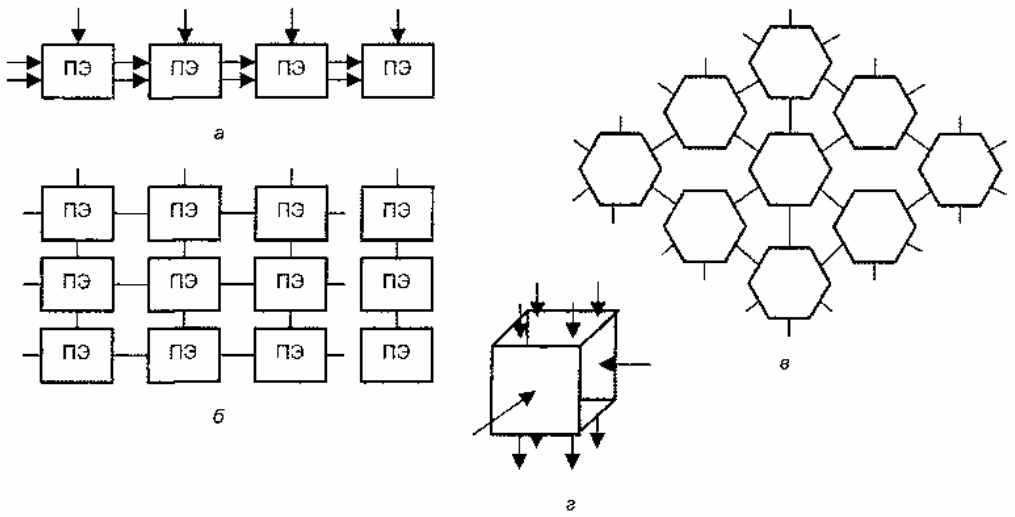


Рисунок 2.19 - Конфигурация систолических матриц:
а) линейная; б) прямоугольная; в) гексагональная; г) трехмерная

Для решения сложных задач конфигурация систолической структуры может представлять собой набор отдельных матриц, сложную сеть взаимосвязанных матриц либо обрабатывающую поверхность. Пол обрабатывающей поверхностью понимается бесконечная прямоугольная сетка ПЭ. где каждый ПЭ соединяется со своими четырьмя соседями (пли большим числом ПЭ). Одним и.ч наиболее подходящих элементов для реализации обрабатывающей поверхности является матрица простых ПЭ или транспьютеров.

Учитывая то, что матрицы ПЭ обычно реализуются на основе сверхбольших интегральных схем, возникающие при этом ограничения привели к тому, что наиболее распространены матрицы с одним, двумя и тремя трактами данных и с одинаковым либо противоположным направлением передачи, обозначаемые как ULA, BLA и TLA соответственно.

ULA (Unidirectional Linear Array) - это однонаправленный линейный процессорный массив, где потоки данных перемещаются в одном направлении. ПЭ в массиве могут быть связаны одним, двумя или тремя трактами. При реализации алгоритма векторного произведения матриц один из потоков данных перемещается вправо, в то время как второй резидентно расположен в массиве (рис. 2.20). Используемый ПЭ представляет собой модифицированный IPS-элемент, поскольку имеется только один тракт данных, а элементы второго потока хранятся в ПЭ массива.

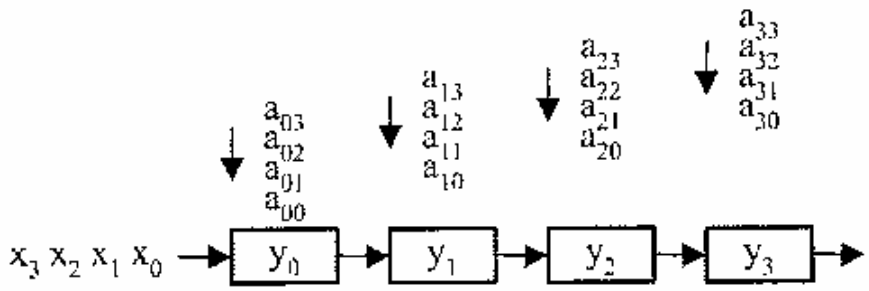


Рисунок 2.20 - Поток данных при векторном перемножении матриц в ULA (*n = 4*)

BLA (Bidirectional Linear Array) — это двунаправленный линейный процессорный массив, в котором два потока данных движутся навстречу друг другу. BLA, где один из потоков является выходным, называется *регулярным*.

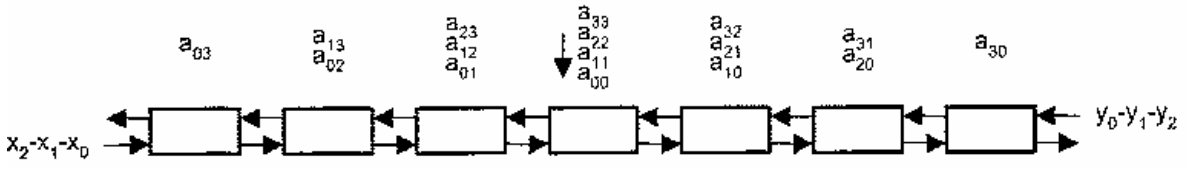


Рисунок 2.21 - Поток данных при векторном перемножении матриц в BLA (*n = 4*)

Реализация рассмотренной ранее операции с применением BLA показана на рис. 2.21. В версии ULA процессоры используются более эффективно, поскольку в них элементы потока следуют в каждом такте, а не через такт, как в BLA.

TLA (Three-path communication Linear Array) — линейный процессорный массив с тремя коммуникационными трактами, в которой по разным направлениям перемещаются три потока данных. На рис. 2.22 показам пример фильтра ARMA, предложенного Кунгом и построенного по схеме TLA. Возможны несколько вариантов такого фильтра, в зависимости от числа выходных потоков данных и от значений, хранящихся в памяти (в примере фигурирует один выходной поток). Процессорные элементы выполняют две операции IPS и обычно называются сдвоенными IPS-элементами. Две версии таких ПЭ представлены на рис. 2.23. ПЭ могут использовать так хранимые в памяти значения (рис. 2.23, а, б), так и внешние данные (рис. 2.23, в, г).

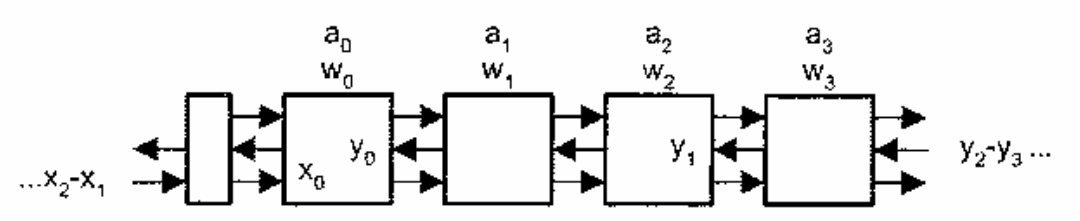


Рисунок 2.22 - Поток данных в TLA фильтра ARMA (*n =3*)

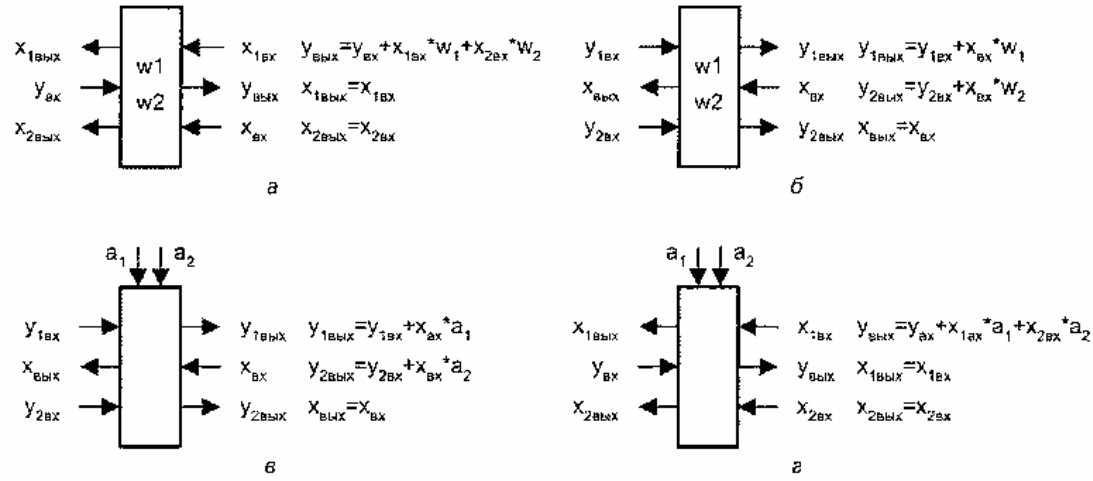


Рисунок 2.23 - Сдвоенные IPS-элементы: а-б) с хранимыми в памяти двумя значениями; в-г) с внешним данными

TLA часто называют сдвоенным конвейером, поскольку он может быть разделен на два линейных конвейера типа BLA. Соответственно, TLA можно получить объединением двух BLA с одним общим потоком данных.

Представленные реализации алгоритма векторного произведения матриц выполняют эту операцию за одно и то же время, но в случае ULA в вычислениях участвуют вдвое меньше процессорных элементов. С другой стороны, ULA использует хранимые в памяти данные, на чтение и запись которых нужно какое-то время. В свою очередь, в схеме BLA требуется дополнительное время на операции ввода/ вывода.

Тип ПЭ выбирается в соответствии с назначением систолической матрицы и структурой пространственных связей. Наиболее распространены процессорные элементы, ориентированные на умножение с накоплением.

На рис. 2.24 показаны ПЭ для двух типов матриц: прямоугольной (см. рис. 2.24, а) и гексагональной (см. рис. 2.24. 6).

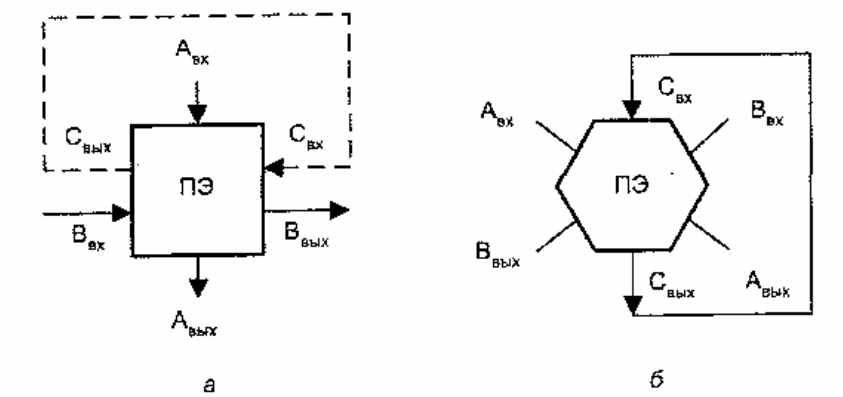


Рисунок 2.24 - Структура ПЭ: а) для прямоугольной систолической матрицы; б) для гексагональной систолической матрицы

В обоих случаях на вход ПЭ подаются два операнда $A_{вх}$, $B_{вх}$, а выходят операнды $A_{вых}$, $B_{вых}$,, и частичная сумма $C_{вых}$. На n-м шаге работы систолической системы ПЭ выполняет операцию

$$C_{вых}^{n} = C_{вх}^{n-1} + A_{вх}^{n-1} * B_{вх}^{n-1}$$

на основе операндов, полученных на (n - 1)-м шаге, при этом операнды на входе и выходе ПЭ одинаковы:

$$A_{вых}^{n} = A_{вх}^{n-1}, B_{вых}^{n-1} = B_{вх}^{n-1}$$

Частичная сумма поступает на вход ПЭ либо с данного процессорного элемента (штриховая линия), либо с соседнего ПЭ матрицы.

2.4.5 Вычислительные системы с командными словами сверхбольшой длины (VLIW)

Архитектура с командными словами сверхбольшой длины или со сверхдлинными командами (VLIW, Very Long Instruction Word) известна с начала 80-х из ряда университетских проектов, по только сейчас, с развитием технологии производства микросхем она нашла свое достойное воплощение. VLIW - это набор команд, организованных наподобие горизонтальной микрокоманды к микропрограммном устройстве управления.

Идея VLIW базируется на том, что задача эффективного планирования параллельного выполнения нескольких команд возлагается на «разумный» компилятор. Такой компилятор вначале исследует исходную программу с целью обнаружить все команды, которые могут быть выполнены одновременно, причем так, чтобы это не приводило к возникновению конфликтов. В процессе анализа компилятор может даже частично имитировать выполнение рассматриваемой программы. На следующем этапе компилятор пытается объединить такие команды в пакеты, каждый из которых рассматривается как одна сверхдлинная команда. Объединение нескольких простых команд в одну сверхдлинную производится по следующим правилам:

- количество простых команд, объединяемых в одну команду сверхбольшой длины, равно числу имеющихся в процессоре функциональных (исполнительных) блоков (ФБ);
- в сверхдлинную команду входят только такие простые команды, которые исполняются разными ФБ, то есть обеспечивается одновременное исполнение всех составляющих сверхдлинной команды.

Длина сверхдлинной команды обычно составляет от 256 до 1024 бит. Такая метакоманда содержит несколько полей (по числу образующих ее простых команд), каждое из которых описывает операцию для конкретного функционального блока. Сказанное иллюстрирует рис. 2.25, где показан возможный формат сверхдлинной команды и взаимосвязь между ее полями и ФБ, реализующими отдельные операции.

Как видно из рисунка, каждое поле сверхдлинной команды отображается на свой функциональный блок, что позволяет получить максимальную отдачу от аппаратуры блока исполнения команд.

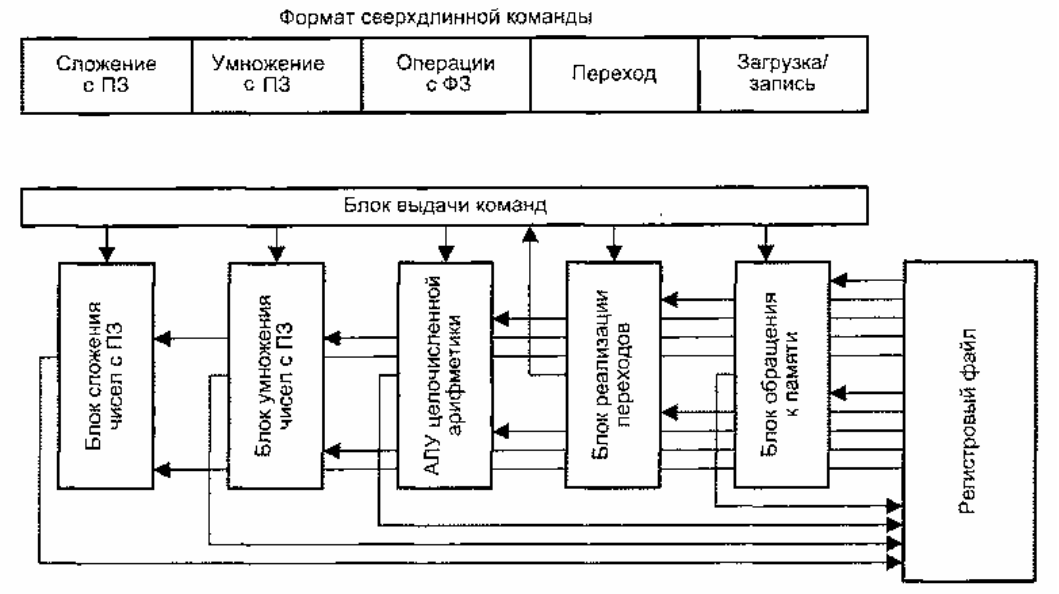


Рисунок 2.25 - Формат сверхдлинной команды и взаимосвязь полей команды

VLIW-архитектуру можно рассматривать как статическую суперскалярную архитектуру. Имеется в виду, что распараллеливание кода производится на этапе компиляции, а не динамически во время исполнения. То, что в выполняемой сверхдлинной команде исключена возможность конфликтов, позволяет предельно упростить аппаратуру VLIW-процессора и, как следствие, добиться более высокого быстродействия.

В качестве простых команд, образующих сверхдлинную, обычно используются команды RISC-типа, поэтому архитектуру VLIW иногда называют постRISC-архитектурой. Максимальное число полей в сверхдлинной команде равно числу вычислительных устройств и обычно колеблется в диапазоне от 3 до 20. Все вычислительные устройства имеют доступ к данным, хранящимся в едином многопортовом регистровом файле. Отсутствие сложных аппаратных механизмов, характерных для суперскалярных процессоров (предсказание переходов, внеочередное исполнение и т. д.) дает значительный выигрыш в быстродействии и возможность более эффективно использовать площадь кристалла. Подавляющее большинство цифровых сигнальных процессоров и мультимедийных процессоров с производительностью более 1 млрд операций/с базируется на VLIW-архитектуре. Серьезная проблема VLIW — усложнение регистрового файла и связей этого файла с вычислительными устройствами.

2.4.6 Потоковые вычислительные системы

Потоковыми называют процессоры, в основе работы которых лежит принцип обработки многих данных с помощью одной команды. Технология SIMD позволяет выполнять одно и то же действие, например вычитание и сложение, над несколькими наборами чисел одновременно. SIMD-операции для чисел двойной точности с плавающей запятой ускоряют работу ресурсоемких приложений для создания контента, трехмерного рендеринга, финансовых расчетов и научных задач. Кроме того, усовершенствованы возможности 64-разрядной технологии MMX (целочисленных SIMD-команд); эта технология распространена на 128-разрядные числа, что позволяет ускорить обработку видео, речи, шифрование, обработку изображений и фотографий. Потоковый процессор повышает общую производительность, что особенно важно при работе с 3D-графическими объектами.

Существуют однопотоковые процессоры (Single-streaming processor — SSP) и многопотоковые процессоры (Multi-Streaming Processor — MSP).

Представителем потоковых процессоров является семейство процессоров Intel начиная с Pentium III, в основе работы которых лежит технология Streaming SIMD Extensions (SSE, потоковая обработка по принципу «одна команда — много данных»). Эта технология позволяет выполнять такие задачи, как обработка речи, кодирование и декодирование видео- и аудиоданных, разработка трех мерной графики и обработка изображений.

Представителями класса SIMD считаются матрицы процессоров: ILLIAC IV, ICL DAP, Goodyear Aerospace MPP, Connection Machine 1 и т. п. В таких системах единое управляющее устройство контролирует множество процессорных элементов. Процессорный элемент получает от устройства управления в каждый фиксированный момент времени команду и выполняет ее над своими локальными данными.

2.5 Организация ВС класса MIMD

Как мы выяснили из классификации Флинна, что кроме архитектуры SIMD, существует и MIMD. Рассмотрим некоторые вычислительные системы этого класса.

2.5.1 Асимметричные мультипроцессорные системы (Asymmetric Multiprocessing - ASMP)

Это архитектура суперкомпьютера, в которой каждый процессор имеет свою оперативную память.

В ASMP используются три схемы (рис. 2.26). В любом случае процессоры взаимодействуют между собой, передавая друг другу сообщения, т. е. как бы образуя скоростную локальную сеть. Передача сообщений может осуществляться через общую шину (рис. 2.26, *а* см. также *MPP-архитектура)* либо благодаря межпроцессорным связям. В последнем случае процессоры связаны либо непосредственно (рис. 2.26, *б),* либо через друг друга (рис. 2.26, *в).* Непосредственные связи используются при небольшом числе процессоров.

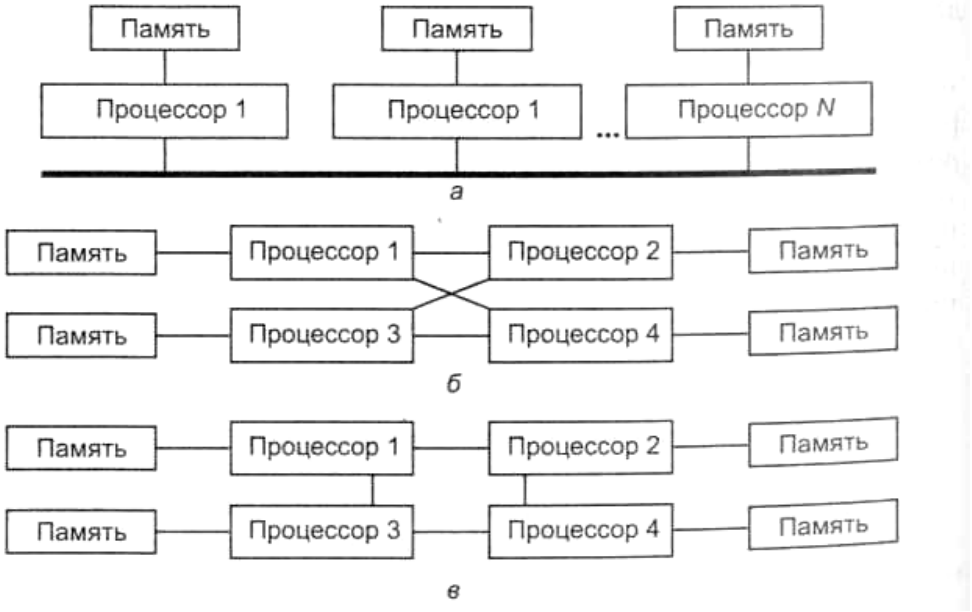


Рисунок 2.26 - Схемы асимметричной многопроцессорной архитектуры

Каждый процессор имеет свою, расположенную рядом с ним оперативную память. Благодаря этому, если это необходимо, процессоры могут располагаться в различных, но рядом установленных корпусах. Совокупность устройств в одном корпусе именуется кластером. Пользователь, обращаясь к кластеру, может работать сразу с группой процессоров. Такое объединение увеличивает скорость обработки данных и расширяет используемую оперативную память. Резко возрастает также отказоустойчивость, ибо кластеры осуществляют резервное дублирование данных. Созданная таким образом система называется кластерной. В этой системе, в

соответствии с ее структурой, может функционировать несколько копий операционной системы и несколько копий прикладной программы, которые работают с одной и той же базой данных (БД), решая одновременно разные задачи.

2.5.2 Симметричные мультипроцессорные системы (Symmetric Multiprocessing - SMP)

Термин SMP (SMP, Symmetric Multiprocessor) относится как к архитектуре ВС, так и к поведению операционной системы, отражающему данную архитектурную организацию. SMP можно определить как вычислительную систему, обладающую следующими характеристиками:

- имеются два или более процессоров сопоставимой производительности;
- процессоры совместно используют основную память и работают в едином виртуальном и физическом адресном пространстве;
- все процессоры связаны между собой посредством шины или по иной схеме, так что время доступа к памяти любого из них одинаково;
- все процессоры разделяют доступ к устройствам ввода/вывода либо через одни и те же каналы, либо через разные каналы, обеспечивающие доступ к одному и тому же внешнему устройству;
- все процессоры способны выполнять одинаковые функции (этим объясняется термин «симметричные»);
- любой из процессоров может обслуживать внешние прерывания;
- вычислительная система управляется интегрированной операционной системой, которая организует и координирует взаимодействие между процессорами и программами на уровне заданий, задач, файлов и элементов данных.

Обратим внимание на последний пункт, который подчеркивает одно из отличий по отношению к слабо связанным мультипроцессорным системам, таким как кластеры, где в качестве физической единицы обмена информацией обычно выступает сообщение или полный файл. В SMP допустимо взаимодействие на уровне отдельного элемента данных, благодаря чему может быть достигнута высокая степень связности между процессами.

Хотя технически SMP-системы симметричны, в их работе присутствует небольшой фактор перекоса, который вносит программное обеспечение. На время

загрузки системы один из процессоров получает статус ведущего (master). Это не означает, что позже, во время работы какие-то процессоры будут ведомыми - все они в SMP-системе равноправны. Термин «ведущий» вводится только затем, чтобы указать, какой из процессоров по умолчанию будет руководить первоначальной загрузкой ВС.

Операционная система планирует процессы или нити процессов (threads) сразу пи всем процессорам, скрывая при этом от пользователя многопроцессорный характер SMP-архитектуры.

На рис. 2.27 в самом общем виде показана архитектура симметричной мультипроцессорной ВС.

Типовая SMP-система содержит от двух до 32 идентичных процессоров, в качестве которых обычно выступают недорогие RISC-процессоры, такие, например, как DEC Alpha. Sun SPARC, MIPS или HP PA-RISC. В последнее время наметилась тенденция оснащения SMP-системами также и CISC-процессорами, в частности Pentium или AMD.

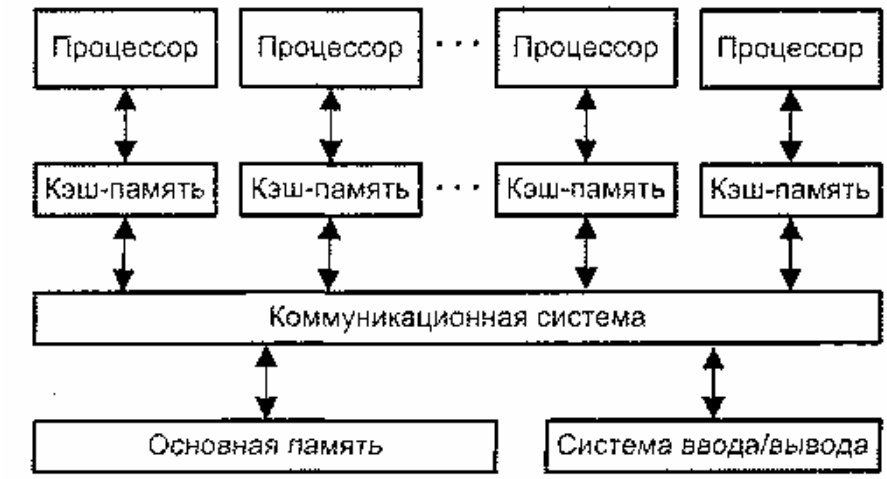


Рисунок 2.27 – Организация симметричной мультипроцессорной ВС

Каждый процессор снабжен локальной кэш-памятью, состоящей из кэш-памяти первого (L1) и второго (L2) уровней. Согласованность содержимого кэш-памяти всех процессоров обеспечивается аппаратными средствами. В некоторых SMP-системы (рис. 2.28). К сожалению, этот прием технически и экономически оправдан лишь,

если число процессоров не превышает четырех. Применение обшей кэш-памяти сопровождается повышением стоимости и снижением быстродействия кэш-памяти.

Все процессоры ВС имеют равноправный доступ к разделяемым основной памяти и устройствам ввода/вывода. Такая возможность обеспечивается коммуникационной системой. Обычно процессоры взаимодействуют между собой через основную память (сообщения и информация о состоянии оставляются в области общих данных). В некоторых SMP-системах предусматривается также прямой обмен сигналами между процессорами.

Память системы обычно строится по модульному принципу и организована так, что допускается одновременное обращение к разным се модулям (банкам). В некоторых конфигурациях в дополнение к совместно используемым ресурсам каждый процессор обладает также собственными локальной основной памятью и каналами ввода/вывода.

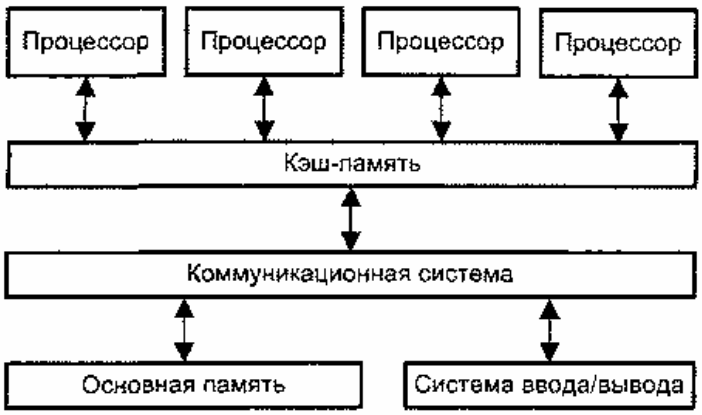


Рисунок 2.28 - SMP-система с совместно используемой кэш-памятью

Важным аспектом архитектуры симметричных мультипроцессоров является способ взаимодействия процессоров с общими ресурсами (памятью и системой ввода/вывода). С этих позиций можно выделить следующие виды архитектуры SMP-систем: с общей шиной и временным разделением; с коммутатором типа «кроссбар»; с многопортовой памятью; с централизованным устройством управления.

Структура и интерфейсы общей шины в основном такие же, как и в однопроцессорной ВС, где шина служит для внутренних соединений (рис. 2.29).

Достоинства и недостатки систем коммуникации на базе общей шины с разделением времени достаточно подробно обсуждались ранее. Применительно к SMP-

системам отметим, что физический интерфейс, а также логика адресации, арбитража и разделения времени остаются теми же, что и в однопроцессорных системах.

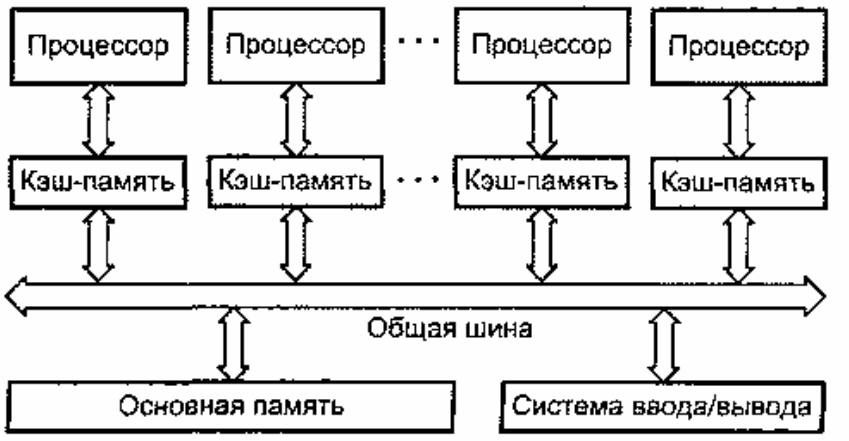


Рисунок 2.29 - Структура SMP-системы с общей шиной

Общая шина позволяет легко расширять систему путем подключения к себе большего числа процессоров. Кроме того, напомним, что шина — это, по существу, пассивная среда, и отказ одного из подключенных к ней устройств не влечет отказа всей совокупности.

В то же время SMP-системам на базе общей шины свойственен и основной недостаток шинной организации — невысокая производительность: скорость системы ограничена временем цикла шины. По этой причине каждый процессор снабжен кэш-памятью, что существенно уменьшает число обращений к шине. Наличие множества кэшей порождает проблему их когерентности, и это одна из основных причин, по которой системы на базе общей шины обычно содержат не слишком много процессоров. Так, в системах Compaq AlphaServer GS140 и 8400 используется не более 14 процессоров Alpha 21264. SMP-система HP N9000 в максимальном варианте состоит из 8 процессоров PA-8500, а система SMP Thin Nodes для RS/6000 фирмы IBM может включать в себя от двух до четырех процессоров PowerPC 604.

Архитектура с коммутатором типа «кроссбар» (рис. 2.30) ориентирована на модульное построение общей памяти и призвана разрешить проблему ограниченной пропускной способности систем с общей шиной.

Коммутатор обеспечивает множественность путей между процессорами и банками памяти, причем топология связей может быть как двумерной, так и трехмерной. Результатом становится более высокая полоса пропускания, что позволяет строить

SMP-систсмы, содержащие больше процессоров, чем в случае общей типы. Типичное число процессоров в SMP-спстемах на базе матричного коммутатора составляет 32 пли G1 Отметим, что выигрыш в производительности достигается, лишь когда разные процессоры обращаются к разным банкам памяти.

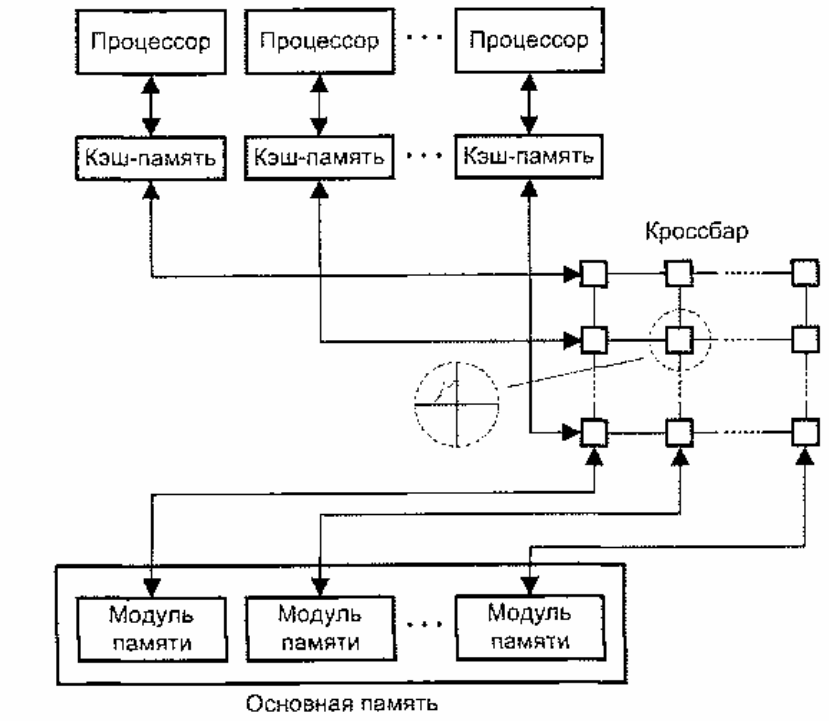


Рисунок 2.30 - Структура SMP-системы с коммутатором типа «кроссбар»

По логике кроссбара строится и взаимодействие процессоров с устройствами ввода/вывода.

В качестве примера ВС с рассмотренной архитектурой можно привести систему Enterprise 10000, состоящую из 64 процессоров, связанных с памятью посредством матричного коммутатора Gigaplane-XB фирмы Sun Microsystems (кроссбар 1G x 16). В IBM RS/6000 Enterprise Server Model S70 коммутатор типа «кроссбар» обеспечивает работу 12 процессоров RS64. В SMP-системах ProLiant 8000 и 8500 фирмы Compaq для объединения с памятью и между собой восьми процессоров Pentium III Xeon применена комбинация нескольких шин и кроссбара.

Концепция матричного коммутатора (кроссбара) не ограничивается симметричными мультипроцессорами. Аналогичная структура связей применяется для объединения узлов в ВС типа CC-NUMA и кластерных вычислительных системах.

Многопортовая организация запоминающего устройства обеспечивает любому процессору и модулю ввода/вывода прямой и непосредственный доступ к банкам основной памяти (ОП). Такой подход сложнее, чем при использовании шипы, поскольку требует придания ЗУ основной памяти дополнительной, достаточно сложной логики. Тем не менее это позволяет поднять производительность, так как каждый процессор имеет выделенный тракт к каждому модулю ОП. Другое преимущество многопортовой организации - возможность назначить отдельные модули памяти в качестве локальной памяти отдельного процессора. Эта особенность позволяет улучшить защиту данных от несанкционированного доступа со стороны других процессоров.

Централизованное устройство управления (ЦУУ) сводит вместе отдельные потоки данных между независимыми модулями: процессором, памятью, устройствами ввода/вывода. ЦУУ может буферизировать запросы, выполнять синхронизацию и арбитраж. Оно способно передавать между процессорами информацию о состоянии и управляющие сообщения, а также предупреждать об изменении информации в кэшах. Недостаток такой организации заключается в сложности устройства управления, что становится потенциальным узким местом в плане производительности. В настоящее время подобная архитектура встречается достаточно редко, но она широко использовалась при создании вычислительных систем на базе машин семейства IBM 370.

2.5.3 Системы с массовой параллельной обработкой (MPP)

Основным признаком, по которому вычислительную систему относят к архитектуре к массовой параллельной обработкой (MPP, Massively Parallel Processing), служит количество процессоров n. Строгой границы не существует, но обычно при $n>128$ считается, что это уже MPP, а при $n<32$ — еще нет. Обобщенная структура MPP-системы показана на рис. 2.31.

Главные особенности, по которым вычислительную систему причисляют к классу MPP можно сформулировать следующим образом:

- стандартные микропроцессоры;
- физически распределенная память;
- хорошая масштабируемость (до тысяч процессоров);

- асинхронная MIMD-система пересылкой сообщений;
- программа представляет собой множество процессов, имеющих отдельные адресные пространства.

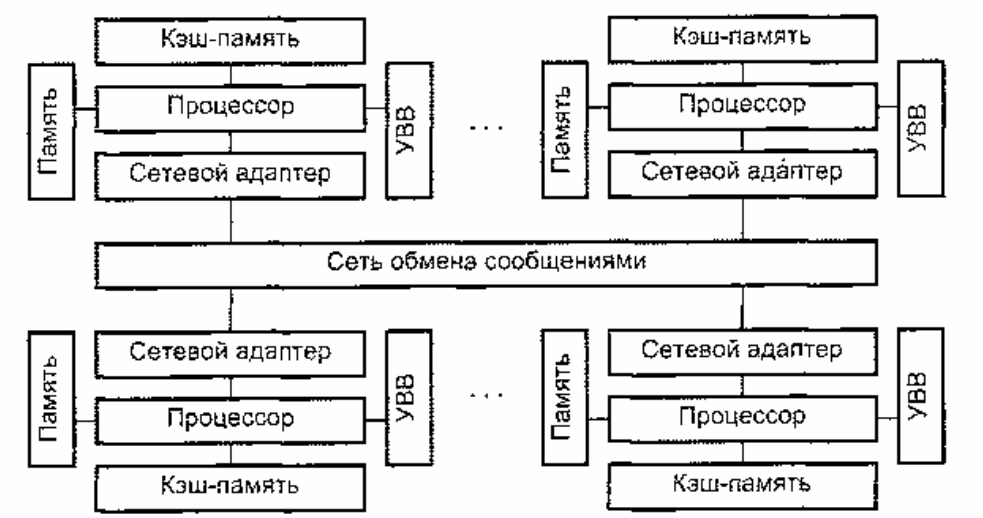


Рисунок 2.31 - Структура вычислительной системы с массовой параллельной обработкой

Основные причины появления систем с массовой параллельной обработкой — это, во-первых, необходимость построения ВС с гигантской производительностью и, во-вторых, стремление раздвинуть границы производства ВС в большом диапазоне, как производительности, так и стоимости. Для МРР-системы, в которой количество процессоров может меняться в широких пределах, всегда реально подобрать конфигурацию с заранее заданной вычислительной мощностью и финансовыми вложениями.

Если говорить о МРР как о представителе класса MIMD с распределенной памятью п отвлечься от организации ввода/вывода, то эта архитектура является естественным расширением кластерной на большое число узлов. Отсюда для МРР-систем характерны все преимущества и недостатки кластеров, причем в связи с повышенным числом процессорных узлов как плюсы, так и минусы становятся гораздо весомее.

Характерная черта МРР-систем - наличие единственного управляющего устройства (процессора), распределяющего задания между множеством подчиненных

ему устройств, чаще всего одинаковых (взаимозаменяемых), принадлежащих одному или нескольким классам. Схема взаимодействия в общих чертах проста:

- центральное управляющее устройство формирует очередь заданий, каждому из которых назначается некоторый уровень приоритета;
- по мере освобождения подчиненных устройств пм передаются задания из очереди:
- подчиненные устройства оповещают центральный процессор о ходе выполнения задания, в частности о завершении выполнения или о потребности в дополнительных ресурсах;
- у центрального устройства имеются средства для контроля работы подчиненных процессоров, в том числе для обнаружения нештатных ситуаций, прерывания выполнения задания в случае появления более приоритетной задачи.

В некотором приближении имеет смысл считать, что на центральном процессоре выполняется ядро операционной системы (планировщик заданий), а на подчиненных ему — приложения. Подчиненность между процессорами может быть реализована как на аппаратном, так и на программном уровне.

Вовсе не обязательно, чтобы MPP-система имела распределенную оперативную память, когда каждый процессорный узел владеет собственной локальной памятью. Так, например, системы SPP1000/XA и SPPI200/XA являют собой пример ВС с массовым параллелизмом, память которых физически распределена между узлами, но логически она общая для всей вычислительной системы. Тем не менее, большинство MPP-систем имеют как логически, так и физически распределенную память.

Благодаря свойству масштабируемости, MPP-системы являются сегодня лидерами по достигнутой производительности; наиболее яркий пример этому — Intel Paragon с 6768 процессорами. С другой стороны, распараллеливание в MPP-системах по сравнению с кластерами, содержащими немного процессоров, становится еще более трудной задачей. Следует помнить, что приращение производительности с ростом числа процессоров обычно вообще довольно быстро убывает (см. закон Амдала). Кроме того, достаточно трудно найти задачи, которые сумели бы эффективны загрузить множество процессорных узлов. Сегодня не так уж много приложений могут эффективно выполняться на MPP-системе. Имеет место также проблема переносимости программ между системами с различной архитектурой. Эффектив-

ность распараллеливания во многих случаях сильно зависит от деталей архитектуры MPP-системы, например топологии соединения процессорных узлов.

Самой эффективной была бы топология, в которой любой узел мог бы напрямую связаться с любым другим узлом, но и ВС на основе MPP это технически трудно реализуемо. Как правило, процессорные узлы в современных MPP-компьютерах образуют или двухмерную решетку (например, в SNI/Pyramid RM1000) или гиперкуб (как в суперкомпьютерах nCube).

Поскольку для синхронизации параллельно выполняющихся процессов необходим обмен сообщениями, которые должны доходить из любого узла системы в любой другой узел, важной характеристикой является диаметр системы D. В случае двухмерной решетки $D \sim sqrt(n)$, и случае гиперкуба $D \sim 1n(n)$. Таким образом, при увеличении числа узлов более выгодна архитектура гиперкуба.

Время передачи информации от узла к узлу зависит от стартовой задержки и скорости передачи. В любом случае, за время передачи процессорные узлы успевают выполнить много команд, и это соотношение быстродействия процессорных узлов и передающей системы, вероятно, будет сохраняться - прогресс в производительности процессоров гораздо весомее, чем в пропускной способности каналов связи. Поэтому инфраструктура каналов связи в MPP-системах является объектом наиболее пристального внимания разработчиков.

Слабым местом MPP было и есть центральное управляющее устройство (ЦУУ) - при выходе его из строя вся система окапывается неработоспособной. Повышение надежности ЦУУ лежит па путях упрощения аппаратуры ЦУУ и/или ее дублирования.

Несмотря на все сложности, сфера применения ВС с массовым параллелизмом постоянно расширяется. Различные системы этого класса эксплуатируются во многих ведущих суперкомпьютерных центрах мира. Следует особенно отметить компьютеры Cray T3D и Cray T3E, которые иллюстрируют тот факт, что мировой лидер производства векторных суперЭВМ, компания Cray Research, уже не ориентируется исключительно на векторные системы. Наконец, нельзя не вспомнить, что суперкомпыютерный проект министерства энергетики США основан на MPP-системе на базе Pentium.

2.5.4 Вычислительные системы на базе транспьютеров

Появление транспьютеров связано с идеей создания различных по производительности ВС (от небольших до мощных массивно-параллельных) посредствен прямого соединения однотипных процессорных чипов. Сам термин объединяет два понятия — «транзистор» и «компьютер».

Транспьютер — это сверхбольшая интегральная микросхема (СБИС), заключающая в себе центральный процессор, блок операций с плавающей запятой (за исключением транспьютеров первого поколения Т212 и Т414), статическое оперативное запоминающее устройство, интерфейс с внешней памятью и несколько каналов связи. Первый транспьютер был разработан в 1986 году фирмой Inmos.

Канал связи состоит из двух последовательных линии для двухстороннего обмена. Оп позволяет объединить транспьютеры между собой и обеспечить взаимные коммуникации. Данные могут пересылаться поэлементно или как вектор. Одна из последовательных линии используется для пересылки пакета данных, а вторая — для возврата пакета подтверждения, который формируется как только пакет данных достигнет пункта назначения.

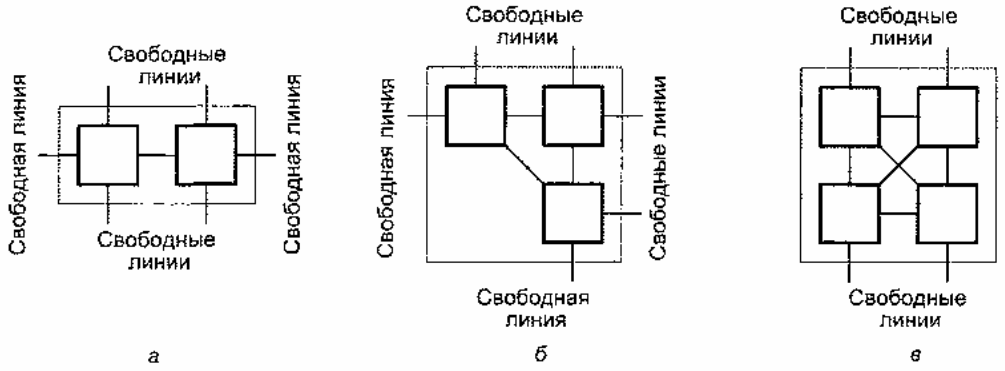


Рисунок 2.32 - Группы из полностью взаимосвязанных транспьютеров: а) два транспьютера; б) три транспьютера; в) четыре транспьютера

На базе транспьютеров легко могут быть построены различные виды ВС. Так, четыре канала связи обеспечивают построение двухмерного массива, где каждый транспьютер связан с четырьмя ближайшими соседями. Возможны и другие конфигурации, например, объединение в группы с последующим соединением групп между собой. Если группа состоит из двух транспьютеров, для подключения ее к группам свободными остаются шесть каналов связи (рис. 2.32, а). Комплекс из трех транспьютеров также оставляет свободными шесть каналов (рис. 2.32, б), а для связи

с «квартетом» транспьютеров остаются еще четыре канала связи (рис. 2.32, в). Группа из пяти транспьютером может иметь полный набор взаимосвязей, но за счет потери возможности подключения к другим группам.

Особенности транспьютеров потребовали разработки для них специального языка программирования Occam. Название языка связано с именем философа-схоласта четырнадцатого века Оккама — автора концепции «бритвы Оккама»: «*entia practer necessitatem non sunl multiplicanda*» — «*понятия не должны умножаться без необходимости*». Язык обеспечивает описание простых операций пересылки данных между двумя точками, а также позволяет явно указать на параллелизм при выполнении программы несколькими транспьютерами. Основным понятием программы на языке Occam является процесс, состоящий из одного пли более операторов программы, которые могут быть выполнены последовательно или параллельно. Процессы могут быть распределены по транспьютерам вычислительной системы, при этом оборудование транспьютера поддерживает совместное использование транспьютера одним или несколькими процессами.

Принято говорить о двух поколениях транспьютеров и языка Occam. Первое поколение отражает требования тех приложений, для которых транспьютеры и разрабатывались: цифровой обработки сигналов и систем реального времени. Для подобных задач нужны сравнительно небольшие ВС со скоростными каналами связи (главным образом, между соседними процессорами) и быстрым переключением контекста. Под контекстом понимается содержимое регистров, которое при переходе к новой задаче в ходе многозадачной обработки может быть изменено и поэтому должно быть сохранено, а при возврате к старой задаче — восстановлено. Многомашинные ВС, построенные на транспьютерах первого поколения (T212, T414 и T805), по своей производительности были сравнимы с другими типами ВС того времени.

С появлением вычислительных систем второго и третьего поколений стало ясно, что ВС на транспьютерах ранней организации уже стали неконкурентоспособными, что и побудило к созданию их второго поколения (T9000). В последних существенно повышена производительность и улучшены каналы связи. Главная особенность транспьютеров второго поколения — развитые коммуникационные возможности, хотя в вычислительном плане, даже несмотря на

наличие в них блоков для операций с плавающей запятой, они сильно уступают универсальным микропроцессорам, таким как PowerPC и Pentium.

Обобщенная структура транспьютера, показанная на рис. 2.33, включает в себя:

- центральный процессор;
- АЛУ для операций с плавающей запятой;
- каналы связи;
- внутреннюю память (ОЗУ);
- интерфейс для подключения внешней памяти;
- интерфейс событий (систему прерываний);
- логику системного сервиса (систему обслуживания);
- таймеры.

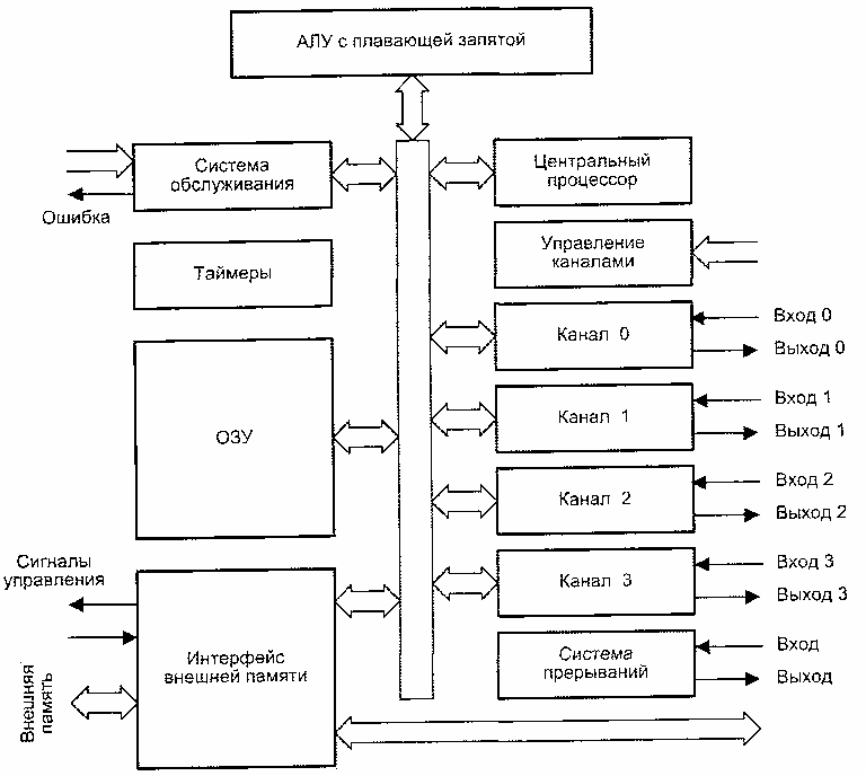


Рисунок 2.33 - Базовая внутренняя архитектура транспьютера

Первый транспьютер Т212 содержал 16-разрядный арифметический процессор. Последующие транспьютеры были оснащены 32-разрядным целочисленным процессором (Т414,1985) и процессором с плавающей запятой (Т800, Т9000), дающим существенное повышение скорости вычислении (до 100 MIPS). Версии, поддерживающие процессор с плавающей запятой, организованы так, что этот процессор и целочисленный процессор могут работать одновременно. В дополнение,

в T9000 добавлена внутренняя кэш-память и процессор виртуального канала. Сам по себе процессор транспьютера построен по архитектуре RISC, имеет микропрограммное устройство управление, а команды в нем выполняются за минимальное число циклов процессора. Простые операции, такие как сложение пли вычитание, занимают одни цикл, в то время как более сложные операции требуют нескольких циклов. Команды состоят из одного или нескольких байтов. Большинство версий транспьютеров имеют по четыре последовательных канала связи со скоростью передачи по каналу порядка 10 Мбит/с. По мере развития транспьютеров повысилась скорость передачи по каналам связи. Емкость внутренней памяти (вначале 2 Кбайт) также возросла. Схема этого интерфейса программируется и способна формировать различные сигналы для удовлетворения различных требований самых разнообразных микросхем внешней памяти.

Передача информации производится синхронно под воздействием либо общего генератора тактовых импульсов (ГТИ), либо локальных ГТИ с одинаковой частотой следования импульсов. Информация передается в виде пакетов. Каждый раз, когда пересылается пакет данных, приемник отвечает пакетом подтверждения (рис. 2.34).

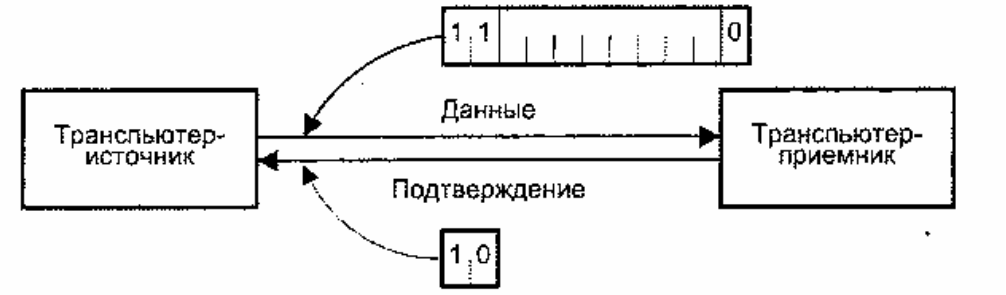


Рисунок 2.34 - Организация ввода/вывода в транспьютерной системе

Пакет данных состоит из двух битов-единиц, за которыми следуют 8-битовые данные и ноль (всего 11 бит). Пакет подтверждения — это простая комбинация 10 (всего два бита), она может быть передана, как только пакет данных будет идентифицирован интерфейсом входного канала. Каналы обеспечивают аппаратную поддержку операторов ввода и вывода языка Occam и функционируют словно каналы прямого доступа к памяти, то есть пакеты могут пересылаться один за другим как векторы. Для коммуникаций между процессами внутри транспьютера вместо внешних каналов операторы ввода/вывода используют внутренние каналы транспьютера.

Интерфейс событий дает возможность внешнему устройству привлечь внимание и получить подтверждение. Этот интерфейс функционирует как входной канал и аналогично программируется.

2.5.5 Гибридная архитектура (NUMA). Организация когерентности многоуровневой иерархической памяти

Главная особенность гибридной архитектуры NUMA (nonuniform memory access) — неоднородный доступ к памяти.

Гибридная архитектура воплощает в себе удобства систем с общей памятью и относительную дешевизну систем с раздельной памятью. Суть этой архитектуры — в методе организации памяти, а именно: память является физически распределенной по различным частям системы, но логически разделяемой, так что пользователь видит единое адресное пространство. Система состоит из однородных базовых модулей (плат), состоящих из небольшого числа процессоров и блока памяти. Модули объединены с помощью высокоскоростного коммутатора. Поддерживается единое адресное пространство, аппаратно поддерживается доступ к удаленной памяти, т. е. к памяти других модулей. При этом доступ к локальной памяти осуществляется в несколько раз быстрее, чем к удаленной. По существу архитектура NUMA является MPP (массивно-параллельной) архитектурой, где в качестве отдельных вычислительных элементов берутся SMP-узлы.

Пример структурной схемы компьютера с гибридной сетью (рис. 2.35): четыре процессора связываются между собой с помощью кроссбара в рамках одного SMP-узла. Узлы связаны сетью типа «бабочка» (Butterfly). Впервые идею гибридной архитектуры предложил и воплотил в системах серии Exemplar Стив Воллох. Вариант Воллоха — система, состоящая из восьми SMP-узлов. Фирма HP купила идею и реализовала на суперкомпьютерах серии SPP. Идею подхватил Сеймур Крей (Seymour R. Cray), добавил новый элемент и воплотил в системах серии Exemplar — когерентный кэш, создав так называемую архитектуру cc-NUMA (Cache Coherent Non-Uniform Memory Access), которая расшифровывается как «неоднородный доступ к памяти с обеспечением когерентности кэшей».

Известны также гибридные структуры с коммутатором (рис. 2.36). Здесь каждый процессор работает со своей памятью, но модули устройств памяти связаны

друг с другом с помощью коммутатора (рис. 2.36, а). Коммутаторы, именуемые также узлами, могут также включаться между группами процессоров (ПР) и модулей памяти (П). Здесь сообщения между процессорами и памятью передаются через несколько узлов (рис. 2.36, б).

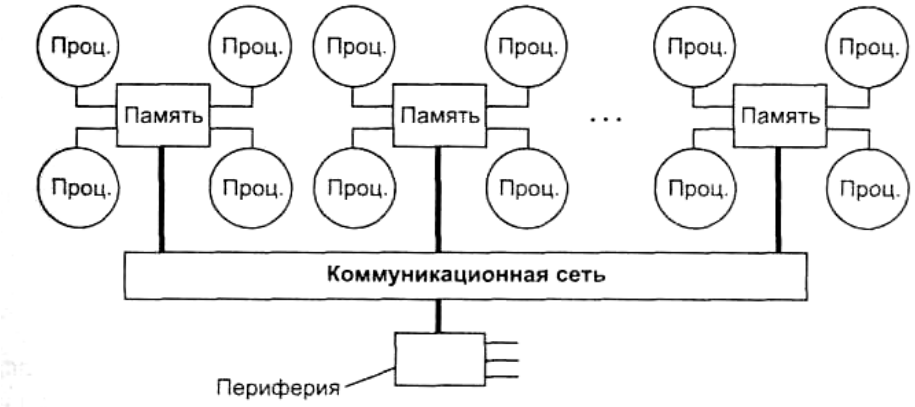


Рисунок 2.35 - Гибридная архитектура

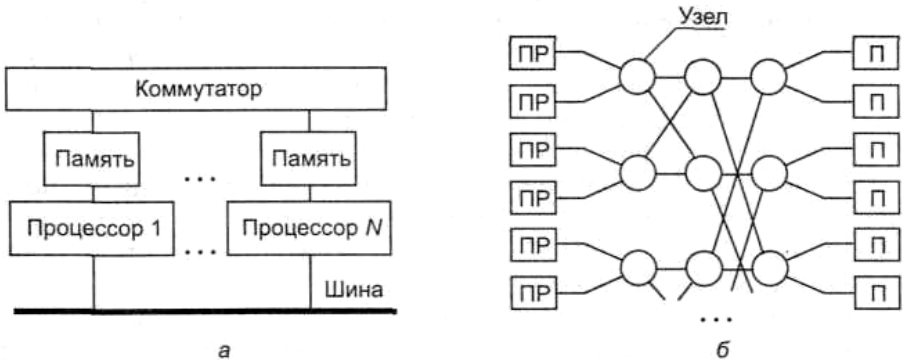


Рисунок 2.36 - а) Гибридная архитектура с коммутатором; б) многокаскадная коммутация

Наиболее известными системами архитектуры cc-NUMA являются: HP 9000 V-class в SCA-конфигурациях, SGI Origin3000, Sun HPC 15000, IBM/Sequent NUMA-Q 2000. На настоящий момент максимальное число процессоров в cc-NUMA-системах может превышать 1000 (серия Origin3000). Обычно вся система работает под управлением единой ОС, как в SMP. Возможны также варианты динамического «расслоения» системы, когда отдельные «разделы» системы работают под управлением разных ОС. При работе с NUMA-системами, также как с SMP, используется парадигма программирования с общей памятью.

2.5.6 Кластерная архитектура

Кластер представляет собой два или более компьютеров (часто называемых узлами), которые объединяются с помощью сетевых технологий на базе шинной архитектуры или коммутатора и предоставляются пользователю в качестве единого информационно-вычислительного ресурса. В качестве узлов кластера могут выступать серверы, рабочие станции или обычные персональные компьютеры. Преимущество кластеризации для повышения работоспособности становится очевидным в случае сбоя какого-либо узла: при этом другой узел кластера может взять на себя нагрузку неисправного узла, и пользователи не заметят прерывания в доступе. Возможности масштабируемости кластеров позволяют многократно увеличивать производительность приложений для большего числа пользователей технологий (Fast/Gigabit Ethernet, Myrinet) на базе шинной архитектуры или коммутатора. Такие суперкомпьютерные системы являются самыми дешевыми, поскольку собираются на базе стандартных комплектующих элементов {«off the shelf»), процессоров, коммутаторов, дисководов и внешних устройств.

Кластеризация может быть осуществлена на разных уровнях компьютерной системы, включая аппаратное обеспечение, операционные системы, программы-утилиты, системы управления и приложения. Чем больше уровней системы объединены кластерной технологией, тем выше надежность, масштабируемость и управляемость кластера.

Условное деление на классы предложено Я. Радаевским и Д. Эдлайном:

- Класс I. Машина строится целиком из стандартных деталей, которые продают многие продавцы компьютерных компонент (низкие цены, простое обслуживание, аппаратные компоненты доступны из различных источников).
- Класс П. Система включает эксклюзивные или не широко распространенные детали. Этим можно достичь очень хорошей производительности, однако при более высокой стоимости.

Как уже указывалось ранее, кластеры могут существовать в различных конфигурациях. Наиболее употребляемыми типами кластеров являются:

- системы высокой надежности;
- системы для высокопроизводительных вычислений;
- многопоточные системы.

Отметим, что границы между этими типами кластеров до некоторой степени размыты, и часто существующий кластер может иметь такие свойства или функции, которые выходят за рамки перечисленных типов. Более того, при конфигурировании большого кластера, используемого как система общего назначения, приходится выделять блоки, выполняющие все перечисленные функции.

Кластеры для высокопроизводительных вычислений предназначены для параллельных расчетов. Эти кластеры обычно собраны из большого числа компьютеров. Разработка таких кластеров является сложным процессом, требующим на каждом шаге аккуратных согласований таких вопросов как инсталляция, эксплуатация и одновременное управление большим количеством компьютеров, технические требования параллельного и высокопроизводительного доступа к одному и тому же системному файлу (или файлам) и межпроцессорная связь между узлами и координация работы в параллельном режиме. Эти проблемы проще всего решаются при обеспечении единого образа операционной системы для всего кластера. Однако реализовать подобную схему удается далеко не всегда, и она обычно применяется лишь для не слишком больших систем.

Многопоточные системы используются для обеспечения единого интерфейса к различным ресурсам, которые могут со временем произвольно наращиваться (или сокращаться) в размере. Наиболее общий пример этого представляет собой группа Web-серверов.

В 1994 г. Т. Стерлинг (Sterling) и Д. Беккер (Becker) создали 16-узловой кластер из процессоров Intel DX4, соединенных сетью Ethernet (10 Мбит/с) с дублированием каналов. Они назвали его «Beowulf» по названию старинной эпической поэмы. Кластер возник в центре NASA Goddard Space Flight Center для поддержки необходимыми вычислительными ресурсами проекта Earth and Space Sciences. Проектно-конструкторские работы над кластером быстро превратились в то, что известно сейчас под названием проект Beowulf.

Проект стал основой общего подхода к построению параллельных кластерных компьютеров и описывает многопроцессорную архитектуру, которая может с успехом использоваться для параллельных вычислений. Beowulf-кластер, как правило, является системой, состоящей из одного серверного узла (который обычно называется головным узлом), а также одного или нескольких подчиненных узлов

(вычислительных узлов), соединенных посредством стандартной компьютерной сети. Система строится с использованием стандартных аппаратных компонент, таких, как ПК, стандартных сетевых адаптеров (например, Ethernet) и коммутаторов. Не существует особого программного пакета, называемого «Beowulf», вместо этого имеется несколько фрагментов программного обеспечения, которые многие пользователи нашли пригодными для построения кластеров Beowulf (такие программные продукты, как ОС Linux, системы передачи сообщений PVM, MPI, системы управления очередями заданий и другие стандартные продукты). Серверный узел контролирует весь кластер и обслуживает файлы, направляемые к клиентским узлам.

Архитектура кластерной системы (способ соединения процессоров друг с другом) определяет ее производительность в большей степени, чем тип используемых в ней процессоров. Критическим параметром, влияющим на величину производительности такой системы, является расстояние между процессорами. Так, соединив вместе 10 персональных компьютеров, можно получить систему для проведения высокопроизводительных вычислений. Проблема, однако, будет состоять в нахождении наиболее эффективного способа соединения стандартных средств друг с другом, поскольку при увеличении производительности каждого процессора в 10 раз производительность системы в целом в 10 раз не увеличится.

Рассмотрим пример построения симметричной 16-процессор-ной системы, в которой все процессоры были бы равноправны. Наиболее естественным представляется соединение в виде плоской решетки, где внешние концы используются для подсоединения внешних устройств (рис. 2.37).

При таком типе соединения максимальное расстояние между процессорами окажется равным 6 (количество связей между процессорами, отделяющих самый ближний процессор от самого дальнего). Теория же показывает, что если в системе максимальное расстояние между процессорами больше 4, то такая система не может работать эффективно. Поэтому, при соединении 16 процессоров друг с другом плоская схема является неэффективной. Для получения более компактной конфигурации необходимо решить задачу о нахождении фигуры, имеющей максимальный объем при минимальной площади поверхности.

В трехмерном пространстве таким свойством обладает шар. Но поскольку необходимо построить узловую систему, то вместо шара приходится использовать куб (если число процессоров равно 8, рис. 2.38, а) или гиперкуб, если число процессоров больше 8. Размерность гиперкуба будет определяться в зависимости от числа процессоров, которые необходимо соединить. Так, для соединения 16 процессоров потребуется 4-мерный гиперкуб (рис. 2.38, б). Для его построения следует взять обычный 3-мерный куб, сдвинуть в еще одном направлении и, соединив вершины, получить гиперкуб.

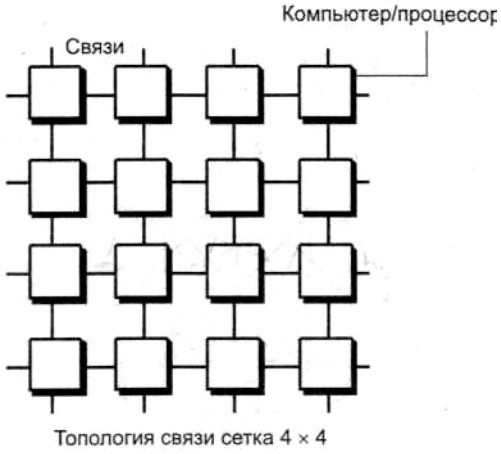


Рисунок 2.37 - Схема соединения процессоров в виде плоской решетки

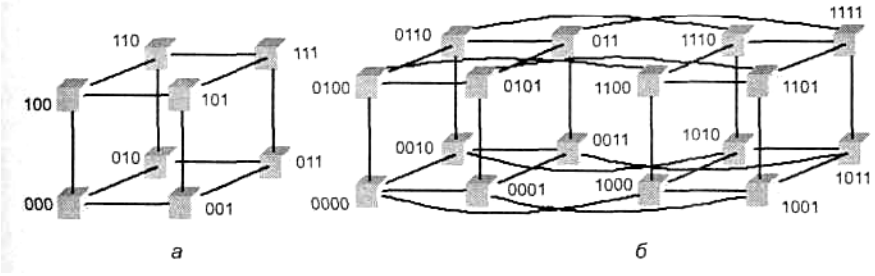


Рисунок 2.38 - Топологии связи: а — трехмерный куб; б — четырехмерный гиперкуб

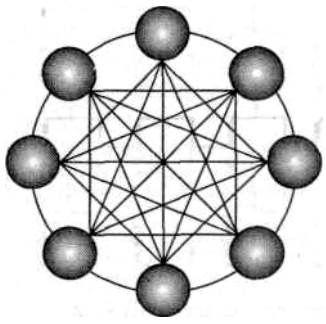


Рисунок 2.39 - Архитектура кольца с полной связью по хордам (Chordal Ring)

Архитектура гиперкуба является второй по эффективности, но самой наглядной. Используются и другие топологии сетей связи: трехмерный тор, «кольцо», «звезда» и другие (рис. 2.39).

Наиболее эффективной считается архитектура с топологией «толстого дерева» (fat-tree). Архитектура «fat-tree» (hypertree) предложена Лейзерсоном (Charles E. Leiserson) в 1985 г. Процессоры локализованы в листьях дерева, в то время как внутренние узлы дерева скомпонованы во внутреннюю сеть (рис. 2.40). Поддеревья могут общаться между собой, не затрагивая более высоких уровней сети.

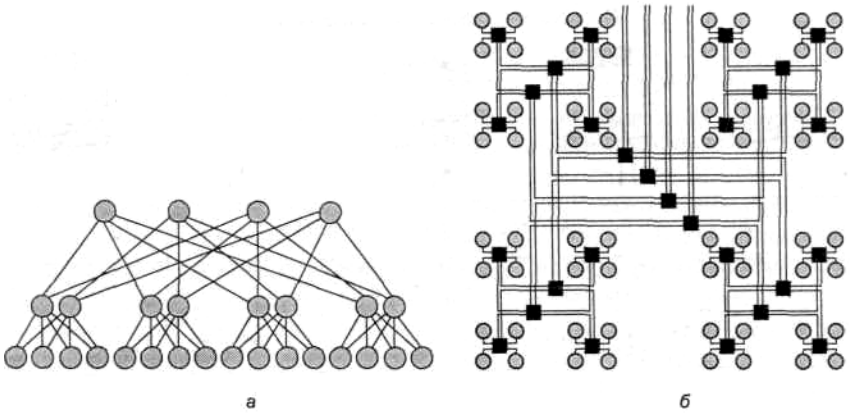


Рисунок 2.40 - Кластерная архитектура «Fat-ree»:
***а* — вид «сбоку»; *б* — вид «сверху»**

Поскольку способ соединения процессоров друг с другом больше влияет на производительность кластера, чем тип используемых в ней процессоров, то может оказаться более рентабельным создать систему из большего числа дешевых компьютеров, чем из меньшего числа дорогих. В кластерах, как правило, используются операционные системы, стандартные для рабочих станций, чаще всего, свободно распространяемые — Linux, FreeBSD, вместе со специальными средствами поддержки параллельного программирования и балансировки нагрузки. При работе с кластерами так же, как и с MPP системами, используют так называемую Massive Passing Programming Paradigm — парадигму программирования с передачей данных (чаше всего — MPI). Дешевизна подобных систем оборачивается большими накладными расходами на взаимодействие параллельных процессов между собой, что сильно сужает потенциальный класс решаемых задач.

3 Перспективные направления в развитии вычислительных систем

3.1 Нанокомпьютеры

Существующая технологическая база производства ЭВМ имеет предел своего развития, причем, по прогнозам разработчиков, этот предел будет достигнут в ближайшие годы. Дальнейшие разработки в области построения ЭВМ и ВС направлены на исследования новых физических явлений и создание новых информационных технологий. Прорыв в области создания новых ЭВМ и ВС связывают в первую очередь с *нанотехнологией. Нанотехнология (от греческого «нанос» — карлик) — термин, используемый для описания процессов, происходящих в пространстве, с линейными размерами от 0,1 до 100 нм (1 нм =10 $^{-9}$ м).* По мнению одного из создателей водородной бомбы и американской программы СОИ Э. Теллера: «*Тот, кто раньше овладеет нанотехнологией, займет ведущее место в техносфере следующего столетия*».

Термин «нанотехнология» был предложен в 1974 г. японским исследователем Танагучи.

В 1997-1998 гг. агентством DARPA (Defense Advanced Research Projects Agency) США была разработана программа формирования новой наукоемкой технологии, получившей название *молекулярной электроники*, или сокращенно *молетроники (moletronics*). Позже появились и другие термины из этой области, такие, как *бионика, биоинженерия*.

Целью программы DARPA было создание функциональных электронных устройств на основе нанотехнологии. При этом все создаваемые логические элементы и элементы памяти должны функционировать в привычном диапазоне температур, иметь высокую плотность и малую мощность электропотребления. Создаваемая память должна быть энергонезависимой, организована по байтному принципу, и иметь терабайтный объем.

Сами функциональные устройства должны создаваться по принципу направленной самоорганизации, а их структура должна быть трехмерной.

Одним из условий программы было то, чтобы все разрабатываемые устройства в какой-то мере «вписывались» в существующие производства электронной аппаратуры и не требовали его полной реконструкции.

Еще не став практической реальностью, нанотехнология «родила» и новые понятия: *квантовые точки*, *квантовые диполи*, *квантовые кучки*, *квантовые вычисления (КВ)*, *квантовые компьютеры (КК)*, *нанотрубки*. Переход к нанотехнологии, несомненно, означает новый этап промышленный революции.

К настоящему времени существует целый ряд разработок, основывающихся на элементах нанотехнологии. Например, компания IBM представила новую технологию хранения информации, с помощью которой можно будет добиться плотности записи порядка триллиона бит на квадратный дюйм. О своем достижении в данной области сообщила и компания HP. В ее лаборатории достигли наивысшей плотности на данный момент и создали демонстрационную 64-битную «микросхему» (скорее, «наносхему») энергонезависимой памяти, в которой роль ячеек памяти играют отдельные молекулы. Этот чип умещается на площади в один квадратный микрон. Кроме того, HP удалось совместить запоминающие и управляющие элементы в одном молекулярном устройстве. По заявлению ее руководителей, у компании HP уже разработана опытная методика производства нанолитографической печати, позволяющая делать копии чипов на пластинах, подобно тому как делаются копии страниц с оригинал-макета в типографиях.

Небольшая компания Nantero сообщила о создании нового экспериментального образца электронной памяти на базе углеродных **нанотрубок.** Инженерам удалось разместить на кремниевой пластине стандартного размера 10 млрд ячеек памяти, каждая из которых состоит из нескольких нанотрубок. Углеродные на-нотрубки (Single-Walled carbon NanoTubes — SWNT) были открыты в 1991 г. ученым С. Ижима (Sumio Iijima, NEC, Япония). Позже было выяснено, что данные образования имеют двойственную природу: они могут себя вести как проводники или как полупроводники, становящиеся проводниками при подаче напряжения определенной величины, при условии закручивания молекулы в спираль.

Нанотрубка — это бесшовный цилиндр, созданный из пленки, сформированной из атомов углерода. Шестиугольные молекулы из атомов углерода получаются из молекул CO (угарный газ) в присутствии катализатора при высокой температуре. Следует отметить, что нанотрубка на порядок прочнее стали и в то же время в 6 раз легче (см. рис. 3.1).

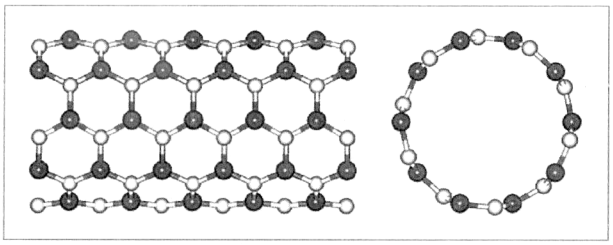


Рисунок 3.1 – Модель нанотрубки BeO (белые атомы – Be, черные - O)

Для производства памяти используется стандартный фотолитографический процесс: вначале на подложку из оксида кремния наносится множество нанотрубок, а в ходе дальнейшей обработки неправильно ориентированные трубки удаляются.

Схема памяти представляет собой две пластинки из оксида кремния, расположенные одна на другой на расстоянии около 100 нм. Нанотрубки как бы «подвешены» на верхней пластинке. При подаче на нижнюю пластинку тока трубки меняют свое положение, соединяя две пластинки. Это состояние соответствует наличию в ячейке бита со значением 1. Если же трубка не замыкает пластин, то в ячейке находится бит со значением 0.

Положение нанотрубки определяется влиянием сил Ван дер Ваальса, которые действуют независимо от наличия электропитания. Электрический импульс нужен лишь для изменения положения трубок. При этом на переключение требуется около 0,5 не против единиц наносекунд у современной оперативной памяти (подобная память получила аббревиатуру **NRAM).** Плотность записи информации в ячейки NRAM постоянно увеличивается и может достичь триллиона бит на квадратный сантиметр, что на несколько порядков больше, чем в случае современной оперативной памяти.

Впрочем, до выхода новой памяти на рынок еще далеко. Углеродные нанотрубки являются все еще экзотическим и дорогостоящим материалом, а производство NRAM, хотя и базируется на традиционной фотолитографии, требует освоения в

промышленности. Однако в перспективе NRAM может оказаться востребованным компьютерным рынком для решения проблемы хранения данных.

Сегодня уже находиться в разработке новое поколение магнитных дисков - устройств температурно-контролируемой записи. В этих системах запись информации будет производиться на пространственно-упорядоченные частицы *FePt* размером всего 5 нм с помощью локального разогрева отдельных частиц фокусированным лазерным пучком в магнитном поле (рис. 3.2). Эта технология была предложена компанией Seagate и на сегодняшний день является «вершиной» нанотехнологического прогресса в области магнитных материалов [4].

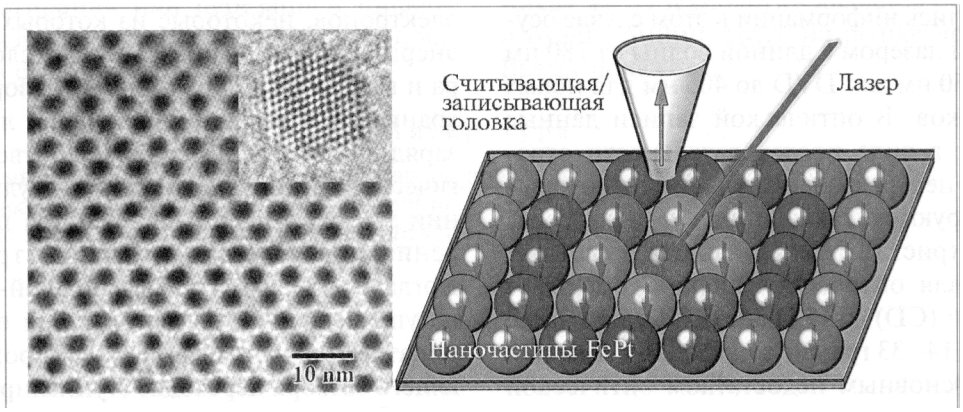


Рисунок 3.2 - Пространственно-упорядоченные массивы наночастиц FePt и принцип температурно-контролируемой записи

Успехи нанотехнологии породили целый ряд проектов, связанных с разработкой компьютеров и систем на их основе, использующих физические принципы, отличные от тех, что применялись при создании классических компьютеров. Среди этих проектов можно отметить работы, связанные с созданием молекулярных, оптических, квантовых, криогенных и нейрокомпьютеров.

3.2 Оптический компьютер

Под термином «*оптический компьютер*» обычно понимают устройство обработки информации с использованием света.

Говоря об отличительных особенностях света как электромагнитной волны, нужно отметить, что частота световой волны на порядок выше частоты электрических сигналов и волн, используемых в современной компьютерной технике. Потому с помощью световой волны в фиксированный интервал времени можно передавать

большее количество информации. Кроме того, поскольку длина световой волны мала, то имеется возможность обработки информации с высокой скоростью.

Следует отметить, что в компьютерах традиционной архитектуры широко используются оптические явления.

В подсистеме ввода-вывода применяются оптические устройства — приводы оптических дисков, сканеры и лазерные принтеры. В подсистеме связи и передачи информации используется оптическое волокно. Передача информации по оптическому волокну заключается в распространении по нему света, при этом скорость передачи на порядок превосходит скорость передачи электрического сигнала. Сочетание оптических и электронных устройств образует **оптоэлектронную** систему.

Первые работы, относящиеся к оптическим или оптоэлектронным ЭВМ, появились в 50-х годах XX в. Интерес к этим ЭВМ был обусловлен целым рядом преимуществ, которыми они обладали по сравнению с традиционными (классическими) ЭВМ. Среди этих преимуществ следует отметить возможность параллельной обработки информации и осуществления сложных двумерных операций типа комплексного преобразования Фурье, корреляции и свертки. Оптические лучи не воздействуют друг на друга, не создают ни коротких замыканий, ни перекрестных помех. Это обеспечивает очень высокую плотность оптических соединений больших массивов оптических логических элементов, недостижимую в системах электронных СБИС. Наиболее эффективно эти ЭВМ можно использовать при обработке изображений, поскольку для таких ЭВМ изображение является естественным двумерным операндом, а основные операции обработки изображений — преобразование Фурье и пространственная фильтрация — осуществляются за один такт работы машины. Исходя из этого, сразу же была определена возможная сфера применения таких ЭВМ. Предполагалось, что они будут использоваться в задачах распознавания и идентификации образов, анализа изображений земной поверхности, полученных при аэрофотосъемках, анализе движения облаков и воздушных масс по изображениям, полученным с метеорологических спутников и др.

Исходя из состояния технической базы, первые оптические ЭВМ пытались создавать с использованием таких источников света, как ртутные лампы и даже солнечный свет. Однако такие попытки оказались тщетными, и интерес к оптическим ЭВМ стал затихать. Так, например, компания IBM на создание оптической ЭВМ

истратила более 100 млн долларов, но в конце концов прекратила работы в этом направлении.

Вновь интерес к оптическим ЭВМ возник в 60-х годах после появления первых промышленных лазеров. Однако и тогда не удалось создать полностью оптическую цифровую ЭВМ. Были предприняты попытки создать оптоэлектронную *гибридную вычислительную систему (ГВС)*.

Гибридная ВС включала в свой состав *когерентный аналоговый оптический процессор*, выполнявший основные операции по параллельной обработке двумерного потока данных, и *цифровой электронный процессор*, который обеспечивал цифровую обработку, ввод-вывод данных и программное управление процессом обработки.

В оптический процессор информация могла поступать из видеоканала, кинопленки, аналоговой памяти, оперативной или внешней памяти, электронного процессора. Входная информация загружалась в специальное устройство ввода, которое представляло собой матричный пространственный модулятор света. Далее информация поступала в оптическую систему процессора для фильтрации. Операционные фильтры могли быть записаны на обычную фотопленку или оперативный носитель в виде библиотеки стандартных фильтров или же синтезированы в процессе обработки информации. Библиотека стандартных фильтров записывалась в виде двумерной матрицы подобно матрице фурье-голограмм.

Спектр входного сигнала направлялся на нужный операционный фильтр с помощью дефлектора, управлявшего излучением лазера. В спектральной плоскости оптического процессора могло храниться до 10^2 и более операционных фильтров.

Съем информации осуществлялся как в выходной плоскости оптического процессора, так и в ее спектральной плоскости. Для этой цели использовались интегральные фотоматрицы. Информация с выхода этих устройств передавалась на электронный процессор для логического анализа, выработки дальнейших управляющих сигналов и отображения результатов. Результаты обработки могли быть переданы в аналоговую или цифровую память, отображены на дисплее или выведены на печать.

При практической реализации предложенной схемы главная трудность заключалась в отсутствии (на тот момент) эффективных и доступных оперативных

устройств ввода информации и синтеза операционных фильтров. Хотя позже и были созданы более современные устройства ввода с оптическим и электрическими входами, ГВС все же не получили широкого распространения. Причиной этого следует считать недостаточный уровень развития технологической базы.

В 1986 г. исследователь из AT&T Д. Миллер создал самый маленький в мире *оптический переключатель* — квадрат со стороной в 10 микрон. Этот переключатель был создан из современных синтетических материалов, обладал скоростью 1 млрд переключений в секунду и при этом не грелся. Миниатюрный переключатель представлял собой зеркальце, на поверхность которого падал лазерный луч. Луч отражался, что соответствовало положению «включено» или 1. Затем на зеркальце падал луч контрольного второго лазера, что делало его неспособным к отражению. Это соответствовало состоянию «выключено» или 0.

Позже были изготовлены симметричные самоэлектрооптические устройства, получившие название *SEED* (SEED — *self-electrooptic device*), работающие в режиме *ИЛИ-НЕ*, и симметричные самоэлектрооптические устройства (S-SEED), работающие в режиме защелки (установка-сброс). Из этих устройств можно было изготавливать сумматоры и другие элементы, применяемые для изготовления ЭВМ. Первым практическим применением оптоэлектронных элементов стали системы, которые работали в последовательном режиме и в которых обрабатываемые данные уже имелись в форме оптических сигналов.

В первой оптической ЭВМ в качестве логических элементов использовались соединители (переключатели) оптических каналов на $LiNbO_3$ (ниобат лития), а в качестве элементов памяти — волоконно-оптические линии задержки. Соединители на $LiNbO_3$ использовались в качестве пятиканальных оптических устройств. Схема такого пятиканального переключателя представлена на рис. 6.3.

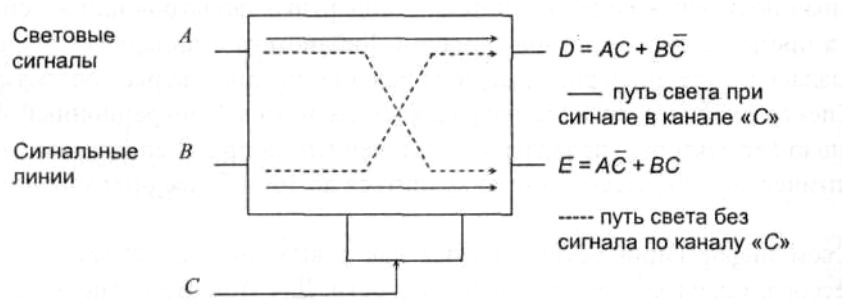


Рисунок 3.3 - Пятиканальный оптический переключатель

Если в канале *C* сигнала нет, то свет, вошедший по каналу *A,* выйдет из канала *E,* а вошедший по каналу *B* — через канал *D.* При поступлении светового сигнала по каналу *C* комбинация «фотодиод-усилитель» выдает сигнал смещения, переводящий устройство в состояние прямого включения. При этом свет, поступающий по каналу *A,* выходит по каналу *D,* а свет, поступающий по каналу *B* — по каналу *E.*

На основе такого переключателя можно построить логические элементы: *И*, *И-НЕ*, *НЕ*, *Буфер*, *ИЛИ* и др.

На рис. 3.4 представлен вариант реализации памяти с использованием линии задержки.

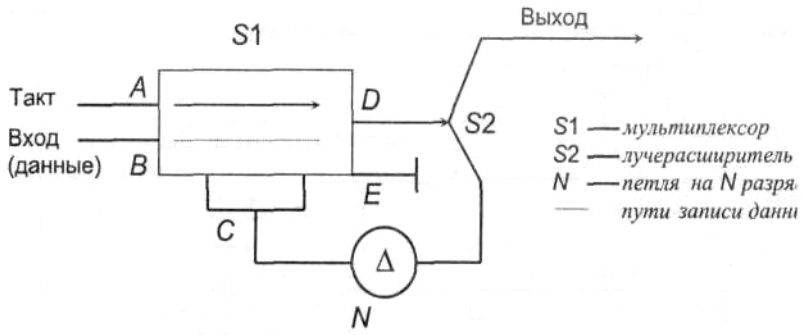


Рисунок 3.4 - Оптическая память на линии задержки

Известно, что число разрядов, умещающихся в петле, определяется соотношением:

$$N = \frac{L}{L_c}, \tag{6.1}$$

где L — длина световодного контура; Lc — расстояние, проходимое оптическим импульсом за один тактовый период.

Например, для размещения 2 Кбайт в оптической памяти с использованием стеклянного волоконно-оптического световода при тактовой частоте 1 ТГц длина петли должна быть равна 3,3 м. Если тактовая частота будет равна 100 МГц, то длина петли увеличится до 3,3 км. Следует заметить, что верхний предел числа разрядов *N,* которые можно записать в петле, определяются тепловыми эффектами. Вследствие потерь в оптических волокнах, коммутирующих и входных устройствах, будет происходить

падение амплитуды сигнала. Поэтому необходимо предусмотреть компенсацию падения амплитуды.

Одним из способов синхронизации и компенсации падения амплитуды является переключение копии тактового сигнала из канала *A* в петлю после прихода сигнала на *SI.*

На рис. 6.5 приведена схема оптического 4-разрядного счетчика.

Работу счетчика можно описать следующим образом. Приход импульса на *SI* переключает его таким образом, что копия тактового сигнала (вход *B)* направляется через выход *E* в волоконно-оптический канал. В расщепителе *S4* импульс расщепляется и, в зависимости от наличия или отсутствия в контуре *M* циркулирующего разряда, направляется или в контур памяти *M*, или в контур переноса *C.* Если в контуре *M* присутствует разряд, то сигнал, поступивший по каналу *C* переключателя *S2*, вызовет переключение импульса на контур переноса и подавление его в переключателе *S3*. Если в контуре Мразряд отсутствует, то импульс подавляется в переключателе *S2* и заводится в контур памяти переключателем *S3*.

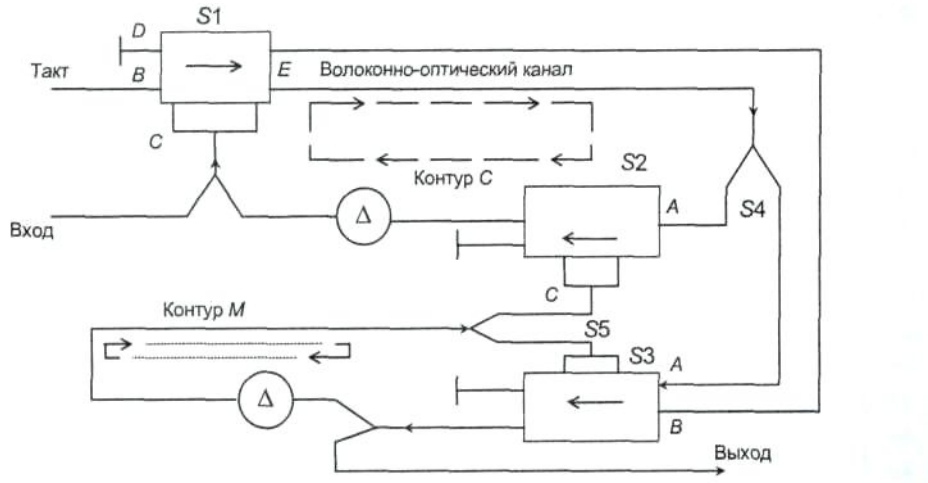


Рисунок 3.5 - Схема 4-разрядного оптического счетчика:

***S1* — демультиплексор; *S2* — схема *И-НЕ*; *S3* — инверсионная выборка; Δ — линия задержки; *C* — контур переноса; *S4*, *S5* — расщепители; *M*— контур памяти**

Важным является вопрос о тактовой частоте, которая зависит не только от частоты повторения лазерных импульсов, но и главным образом от задержек в цепи обратной связи. Применительно к контуру переноса *C* тактовый период не может быть меньше полной задержки при проходе по контуру, поскольку разряд в контуре задержки

должен обойти его и вернуться на переключатель 51 в определенный момент времени, с тем чтобы переключить следующий входящий импульс.

Одна из старейших архитектур, разработанных для оптических компьютеров, архитектура OPLA (Optical Programmable Logic Array) — оптическая программируемая логическая матрица. Многолетние работы по реализации этой архитектуры проводились в Японии. На оптических интегральных схемах была реализована вся булева алгебра. Считалось даже, что компьютеры 5-го и 6-го поколений будут реализованы на оптической элементной базе. Но доведение до стадии коммерчески пригодных продуктов оказалось сложней и длительней, чем ожидалось, и в последнее время сообщений о каких-либо успехах в этом направлении не было.

В 1993 г. в Университете Колорадо профессорами Джорданом и Хейрингом был продемонстрирован оптический компьютер размером с небольшой автомобиль и мощностью недорогого ПК. Оперативная память ОЭВМ представляет собой четырехкилометровые петли оптического волокна. По ним циркулируют импульсы инфракрасного излучения. Четырехметровый импульс кодирует 1, его отсутствие — 0. Кодированные таким образом команды и данные курсируют в линиях задержки, пока управляющий элемент не направит их в процессор. Архитектура получила название *bit-serial architecture {«последовательная битовая архитектура»).* Потоки информации (лучи) коммутируются в процессоре 66 электрооптическими переключателями на $LiNbO_3$.

Несмотря на то, что простые оптические компоненты (переключатели, счетчики, мультиплексоры, демультиплексоры и т. п.) могут работать на тактовых частотах 40 ГГц и более, энергопотребление многоцветных лазеров значительно превышает даже самые «прожорливые» полупроводники на базе арсенида галлия. Нелишне подчеркнуть, что примитивная схемотехника оптических систем все еще оставляет желать лучшего, делая их малопригодными для реализации вычислительных машин общего назначения.

Работы в области создания оптических ЭВМ велись и в бывшем СССР, ведутся и сейчас в России и на Украине (коллектив ученых АН Украины под руководством М. С. Соскина). Большие успехи были достигнуты в использовании явления нелинейного распространения в оптоволоконных линиях. Еще до создания фирмой AT&T своего оптического переключателя, наши ученые уже использовали в подобного рода

переключателях явление низкой нелинейности с удлиненной дистанцией распространения. Это явление было использовано для усиления кумулятивного эффекта взамен применяемого ранее явления высокой нелинейности.

Долгое время в России осуществлялся проект создания оптической сверхвысокопроизводительной вычислительной машины (ОСВМ). Работы по этой теме были начаты еще в 1985 г. в лаборатории академика Г. И. Марчука. В соответствии с проектом оптика в новой ЭВМ должна была использоваться в системах связи и коммутации. Архитектура ОСВМ предусматривала новую организацию вычислительного процесса, структурную надежность и исключение человека из распределения вычислительных ресурсов. ОСВМ должна была дать производительность порядка 10^{15} FLOPS. Однако в отличие от американских разработок эта машина должна была потреблять в десятки раз большую энергию (из-за худших топологических норм) — порядка 10 МВт. Было найдено оригинальное схемотехническое решение, которое исключало задержки распространения сигналов внутри машины и позволило ввести жидкостное охлаждение, раздвинуть блоки и компенсировать высокую рассеиваемую мощность интегральных схем.

В дальнейшем к работе над ОСВМ подключился вновь созданный Институт высокопроизводительных вычислительных систем (ИВВС), директором которого стал академик В. С. Бурцев. К сожалению, в 1998 г. после его ухода работы затормозились.

Первым коммерческим оптическим компьютером считается **Enlight 256** израильской фирмы Lenslet, представленный в 2003 г. Оптический процессор представляет собой цифровой сигнальный процессор с оптическим ускорителем. По сути этот компьютер является гибридным, так как он не полностью оптический, а содержит различные преобразователи света, т. е. ядро компьютера — оптическое, а все остальное — электронное.

Ядро состоит из 256 VCSEL-лазеров, пространственного модулятора света, набора линз и фотоприемников. Производительность процессора составляет 8 TFLOPS, такт — 8 не. За один такт он может перемножить матрицы размером 256 x 256.

К оптоэлектронному компьютеру второго поколения можно отнести и компьютер специального назначения **Zebra True Recognition** корпорации Mytec Technologies, который был представлен в 1995 г. Этот компьютер предназначен для

распознавания дактилоскопических изображений, определения поддельных документов, в том числе страховых полисов, кредитных карточек, а также для идентификации пользователей при входе в корпоративные базы данных. При сравнении отпечатков пальцев он работает сразу со всей картинкой, а не с ее фрагментами, как это делают современные системы.

В заключение заметим, что ближайшее будущее компьютерной техники — за машинами, в которых будут творчески сочетаться компоненты оптики и электроники. Электронные узлы процессора могут быть связаны высокоскоростными световыми волнами информации, которые уже не будут причиной перегрева медных проводов. Электроника хорошо зарекомендовала себя в вычислительных операциях, оптика — в передаче информации. Так что завтрашним днем можно считать оптикоэлектронный компьютерный симбиоз.

3.3 Квантовые компьютеры

Среди проектов, посвященных созданию принципиально новых ЭВМ, наиболее перспективным представляется разработка квантового компьютера. *Квантовый компьютер* будет состоять из компонентов субатомного размера и работать по принципам квантовой механики.

Следует отметить, что уже и в настоящее время существуют множество систем, в которых квантовые эффекты играют существенную роль. В качестве примеров таких систем можно привести различные виды квантовых генераторов (лазеры, мазеры), современные микросхемы, диоды Ганна и др.

Принято считать, что начало работам по квантовым компьютерам положила опубликованная в 1982 г. статья Р. Фейнмана, в которой нобелевский лауреат в области физики обратил внимание на то, что если имеется некоторое *квантовое устройство*, т. е. подчиняющееся законам квантовой механики, то его возможности будут значительно больше возможностей классических компьютеров.

Справедливости ради следует отметить, что еще в 1980 г. советский математик Ю. И. Манин в своих статьях «Вычислимое и невычислимое» и «Доказуемое и недоказуемое» описал сюжет про *квантовые автоматы*, где он говорит о некоторых кардинальных отличиях этих автоматов от классических.

В середине 80-х годов XX в. появились работы Д. Дойча, Е. Бернстайна и У. Вазирини, А. Яо, в которых были описаны модели квантового компьютера Например, была описана модель квантовой машины Тьюринга. Появившаяся в 1994 г. статья П. Шора (P. W. Shor, сотрудник фирмы Lucent Technologies, мате матик) вызвала лавинообразный поток публикаций о *квантовых вычислениях* **(КВ).** П. Шор предложил квантовый алгоритм, позволяющий производить бы струю факторизацию больших чисел. Напомним, что все известные алгоритмы для обычного компьютера — экспоненциальные, т. е. время их вычисления растет как экспонента от числа знаков в записи факторизуемого числа. Так, например, факторизация 129-разрядного числа потребовала бы 500 MIPS-лет или 8 месяцев непрерывной работы системы из 1600 рабочих станций, объединенных через Интернет. А факторизация числа в 500 разрядов потребует времени, превышающего возраст Вселенной, и одновременную работу всех существующих в мире компьютеров. Алгоритм П. Шора, по сравнению с другими, дает многократное ускорение вычислений, причем, чем длиннее факторизуемое число, тем значительнее выигрыш в скорости.

В 1996 г. коллега П. Шора Л. Гровер (L. Grover) предложил квантовый алго ритм быстрого поиска в неупорядоченной базе данных. Этот алгоритм позволяет не только ускорить процесс поиска, но и увеличить примерно в 2 раза число пара метров, учитываемых при поиске оптимальным способом.

Независимо от П. Шора наш ученый А. Китаев (ИТФ им. Л. Д. Ландау) также разработал квантовый алгоритм для решения задачи о дискретном логарифме и некоторых других более общих задач.

Именно середину 90-х годов XX в. можно считать временем рождения новой отрасли вычислений — *квантовых вычислений* **(КВ),** т. е. вычисления на *квантовом компьютере* **(КК).**

Хотя самого КК еще не существует, но уже имеются подходы для его построения.

Прежде чем рассматривать структуры КК, кратко рассмотрим суть КВ. В КВ, как и в обычных компьютерах, оперируют понятием **бит,** которое в этом случае носит название *кубит* **(***qubit, quantum bit***).** Если обычный *бит* — это пространство состояний из двух элементов 0 и 1, то *кубит* — это двумерное комплексное пространство. В квантовой системе можно выполнять только унитарные пре-

образования пространства состояний системы. С точки зрения геометрии такие преобразования — прямой аналог вращений и симметрии обычного трехмерного пространства. Согласно принципу суперпозиции, можно складывать состояния, вычитать их и умножать на комплексное число.

В самом общем случае элементарным шагом при КВ является унитарная операция *L-кубитовой суперпозиции* состояний регистра из L кубитов. Заметим, что при этом выполняется параллельная обработка сразу всех 2^L комплексных амплитуд. Для классической ЭВМ подобная операция потребовала бы 2^L отдельных элементарных шагов. В КВ вычислительный процесс носит характер интерференции, т. е. комплексные амплитуды состояний многих кубитов могут складываться конструктивно и деструктивно.

Кубит — квантовая частица, имеющая два базовых состояния. Этим состояниям могут соответствовать, например, основное и возбужденное состояние атома, направления вверх и вниз спина атомного ядра, два возможных положения электрона и др.

Вне зависимости от принятой структурной схемы основным элементом КК является *квантовый регистр***.** Квантовый регистр — это цепочка кубитов, над которыми можно выполнять одно- и двухбитовые логические операции. К базовым состояниям квантового регистра, образованного L кубитами, так же как и в классическом регистре, относятся все возможные последовательности нулей и единиц длины L. Всего возможны 2^L различных комбинаций. Их можно считать записью чисел в двоичной форме от 0 до $2^L - 1$. Но в отличие от классического регистра, кроме этого, существуют еще и состояния суперпозиции, задаваемые комплексными амплитудами.

Следует заметить, что небольшие квантовые регистры ($L < 20$) могут служить лишь для демонстрации отдельных узлов и принципов работы КК, практической ценности они не имеют. Практически ценный КК должен содержать не менее тысячи кубитов.

Независимо от физической основы КК, принципиальная схема его работы может быть в упрощенном виде описана следующим образом.

До ввода информации в КК все кубиты квантового регистра должны быть приведены в основные *базисные* (булевые) состояния. Эта операция называется *инициализацией***.** Далее каждый кубит подвергается селективному воздействию,

например с помощью импульсов внешнего электромагнитного поля, управляемого классическим компьютером. С помощью этого поля кубиты будут переведены в неосновное состояние, что приведет состояние всего квантового регистра в суперпозицию базисных состояний, задающих бинарное представление числа *n* в виде:

$$n = \sum_{i=1}^{N} n_i * 2^i, \qquad (6.2)$$

На рис. 3.6 представлена структурная схема квантового компьютера. При выборе конкретной схемы КК необходимо решить три принципиальных вопроса.

1. Выбор *элементной базы* (физической среды), которая будет обеспечивать необходимое количество кубитов.

2. Выбор *физического механизма*, определяющего взаимодействие между кубитами.

3. Определение способа *управления кубитами и измерения их состояния* на выходе.

В настоящее время ведутся исследования над двумя различными архитектурами КК: типа клеточного автомата и в виде сети логических элементов. Принципиальных преимуществ ни одна из этих архитектур не дает.

Рисунок 3.6 - Структурная схема квантового компьютера

В качестве *физических явлений*, используемых основой для логических элементов КК, предлагается использовать: взаимодействие одиночных поляризованных фотонов или лазерного излучения с веществом или отдельными атомами; квантовые точки; ядерный магнитный резонанс (ЯМР); объемный спиновой резонанс.

Наиболее реальной на сегодняшний день является реализация КК, использующего объемный спиновой резонанс. В качестве кубитов здесь используются

ориентации ядерных спинов. Работа логических ячеек и запись кубитов осуществляются радиочастотными электромагнитными импульсами со специально подобранными частотой и формой. Основным логическим элементом такого КК будет управляемый инвертор. Ожидается, что уже в ближайшем будущем будет создан КК с разрядностью в 10 кубит.

К настоящему времени определены следующие основные требования, предъявляемые к физической среде, в которой будет реализован КК.

1. Физическая система, представляющая полномасштабный КК, должна состоять из точно известного числа кубитов, число которых должно быть не менее 10^3.

2. Должна быть обеспечена возможность приведения системы в известное начальное состояние (инициализация входного регистра).

3. Необходимо обеспечить высокую степень изоляции КК от внешней среды. Система кубитов должна быть слабо связна с окружением.

4. Необходимо уметь изменять состояние системы в соответствии с заданной последовательностью унитарных преобразований ее фазового пространства.

5. Необходимо обеспечить с достаточно высокой степенью надежности измерение состояний системы на выходе.

Наиболее сложными при реализации являются требования 3 и 4. Рассмотрим в практическом плане те проекты создания КК, которые реализуются в настоящее время.

1. Первый прототип КК, основанный на взаимодействии между заряженными ионами и управлении ими с помощью лазера ИК-диапазона, был предложен австрийскими физиками И. Цираком и П. Цоллером в 1995 г. Реализация этого про екта показала основные недостатки используемого метода: необходимость создания сверхнизких температур, трудности в обеспечении устойчивости состояний ионов в цепочке и ограниченность возможного числа кубитов значением $L < 40$. Тем не менее работы в этом направлении ведутся в Лос-Аламосской лаборатории (LANL) и Национальном институте (NIST) США.

2. В 1997 г. в Массачусетсом Технологическом институте, лаборатории LANL (США) и Оксфорде (Великобритания) были выполнены первые эксперименты по использованию в качестве кубитов атомных ядер с ядерными спинами /= 1/2. Пер воначально для экспериментов использовались молекулы 2, 3-дибромотиофена и

трихлорэтилена. Позднее использовались молекулы цитозина, хлороформа, аланина и других жидкостей с числом спинов-кубитов L = 3, 5, б, 7. Управление процессами осуществлялось с помощью радиочастотных импульсов. Компьютер такого рода получил название ансамблевого КК на основе ЯМР.

3. В 1998 г. Д. В. Аверин предложил использовать в качестве кубитов зарядоые состояния куперовских пар в квантовых точках, связанных переходами Джозефсона. В 1999 г. в Японии (фирма NEC) на этих принципах был создан первый твердотельный кубит.

4. Российский исследователь М. В. Фейгельман (ИТФ им. Л. Д. Ландау) предлагает собирать квантовые регистры из миниатюрных сверхпроводящих колец. Каждое кольцо выполняет роль кубита, а состояния 0 или 1 соответствуют направлению электрического тока в кольце. Переключение кубитов можно осуществлять с помощью магнитного поля.

5. Группа, возглавляемая американским ученым И. Чангом, объявила о сборке 5-битового КК, где в качестве кубитов используются спины ядер некоторых органических молекул.

6. Группа ученых из фирмы IBM и Станфордского университета продемон стрировала семикубитный КК для факторизации чисел по алгоритму Шора. Ком пьютер правильно определил, что делителями числа 15 являются числа 5 и 3. По состоянию на 2001 г. это было самое сложное вычисление, выполненное КК. Ком пьютер представляет собой пробирку с 10^{18} молекул, имеющих 7 ядерных спинов. «Программирование» такого КК осуществляется с помощью электромагнитных импульсов разной частоты. Для получения результата используется ЯМР-сканер.

7. В 2004 г. в Институте компьютерной архитектуры и технологий программирования имени Фраунгофера разработан первый онлайновый эмулятор КК.Установка эмулирует работу 31 кубита и используется для проверки квантовых алгоритмов. Эмулятор представляет собой кластер из 32 узлов на базе процессоров AMD Athlon 3200 с общим объемом ОП 56 Гбайт.

8. В 2005 г. европейское отделение компании Hitachi объявило о создании «базисного элемента» КК. Работы проводились совместно с Кембриджским университетом. Полученный элемент представляет собой микроскопическую частицу размером 50 х 150 нм. Для работы элемента необходима температура, близкая к

абсолютному нулю (-273°С). Полученное состояние кубитов, позволяющее выполнять параллельные вычисления, длилось 200 не, что в 100 раз больше предыдущих результатов. Для сборки полноценного КК необходимо иметь сеть из 10 тыс. таких элементов.

Предполагается, что к 2010 г. появятся первые КК с $L = 100$ кубитов.

3.4 Криогенный компьютер

Под *криогенным компьютером* понимается такой компьютер, элементная база которого основана на явлениях сверхпроводимости и использовании криогенной техники. Понятно, что выделение таких ЭВМ в отдельный класс несколько условно, так как явление сверхпроводимости используется и при разработке других классов и типов ЭВМ.

Следует отметить, что в большинстве разработок, посвященных криогенным ЭВМ, используется *эффект Джозефсона***,** предсказанный в 1962 г. Эффект связан с протеканием сверхпроводящего тока через тонкий слой диэлектрика, разделяющего два сверхпроводника (так называемый контакт Джозефсона).

Рассмотрим основные причины, которые заставляют исследователей заниматься криогенной ЭВМ. Хорошо известно, что быстродействие ЭВМ определяется в первую очередь временем срабатывания ее элементов. Независимо от поколения ЭВМ до сих пор цикл ЭВМ был примерно в 50 раз больше времени срабатывания отдельного элемента (вентиля). Если предположить, что это соотношение сохранится и дальше, то для достижения цикла 1 не необходимо иметь элементы с временем срабатывания 20 пс.

Если в качестве элементной базы ЭВМ использовать полупроводники, то такое быстродействие можно получить, лишь уменьшив длину канала до субмикронных размеров. Этого в принципе можно достичь, используя новые полупроводниковые материалы с высокой подвижностью носителей, охлаждаемые, например, жидким азотом. Однако для получения цикла 1 не недостаточно наличия сверхбыстродействующих элементов, их необходимо еще так упаковать, чтобы наиболее длинный путь сигнал проходил не более, чем за 1 не, а это в лучшем случае 10

см. Из этого следует, что ЭВМ с циклом в 1 не должна иметь объем в несколько кубических дециметров.

Естественно, что в таком объеме трудно сосредоточить многокиловаттную мощность, которую потребляет современная высокопроизводительная ЭВМ, содержащая 10^7-10^8 транзисторов на процессор. Поэтому возникает новая проблема— уменьшение потребляемой мощности. Вот по этому показателю СП-элементы и находятся вне конкуренции. Благодаря их охлаждению до температуры 4,2° К, при которой тепловые потери незначительны, уровень информационных сигналов у них составляет 1 мВ по сравнению с 1 В у транзисторов. Поэтому, если у биполярных транзисторов энергопотребление составляет милливатт на вентиль, у полевых транзисторов — сотни микроватт, то у СП-элементов — единицы микроватт.

Серьезная проблема, возникающая в цифровой полупроводниковой технике, — это межэлементные соединения на кристалле. Линии связи на кристалле определяют все основные параметры и в особенности быстродействие, которое с повышением степени интеграции снижается за счет увеличения средней длины, площади и погонного сопротивления металлизированных проводников. В современных ЭВМ 40% времени цикла теряется на соединениях. Эту проблему также можно решить с использованием СП-элементов, так как СП-полосковая линия является идеальным средством передачи сигналов без искажения и затухания до частоты 10^{12} Гц. Задержка в такой линии составляет 0,08 нс/см, что значительно меньше, чем в обычной линии. Учитывая, что токонесущие возможности СП-линий выше, чем у обычных линий, эти линии можно делать более узкими и размещать более плотно и в меньшем количестве слоев.

Если ориентироваться на уже существующие принципы организации вычислительного процесса, то согласно принципу Ландауэра, при работе в рамках классической логики любое переключение транзистора приводит к выделению тепла, пропорционального температуре транзистора. Если понизить температуру транзистора, можно будет понизить напряжение питания и, следовательно, уменьшить тепловыделение (снижать напряжение питания без снижения температуры процессора нельзя, так как это приведет к сбоям в работе).

Как сильно требуется охладить процессор, чтобы добиться существенного выигрыша в тепловыделении? Из основного соотношения Ландауэра видно, что

охлаждение процессора даже до температуры жидкого азота (77,4° К) не дает больших преимуществ, так как снижает тепловыделение по сравнению с режимом работы при комнатной температуре всего лишь в 4 раза. Если процессор без охлаждения рассеивал, допустим, мощность 60 Вт, то при температуре жидкого азота он будет рассеивать мощность 15 Вт. Охлаждение до температуры жидкого гелия (4,2° К) понижает температуру вычислительного процесса примерно в 100 раз, что дает для рассматриваемого случая мощность рассеяния 600 мВт.

Производительность нанокомпьютера, охлаждаемого жидким гелием, можно оценить следующим образом. Теплота испарения жидкого гелия примерно равна $3\text{-}10^3$ Дж/л. Таким образом, одного кубического миллиметра жидкого гелия, расходуемого за 1 секунду при температуре 4,2° К, будет достаточно для отвода ланда-уэровского тепла от машины с вычислительной производительностью примерно 5-Ю^{19} MIPS. Если предположить, что одновременно будут переключаться 100 млн одноэлектронных транзисторов, то рабочая частота нанокомпьютера может быть выше 100 ГГц, а тепловыделение — лишь 3 мВт. Создание криогенного наночи-па — дело вполне реальное, так как в системе замкнутого оборота криогенного кулера должно быть всего несколько кубических миллиметров жидкого гелия.

При широком коммерческом производстве гелиевые кулеры для ПК будут размером не более воздушных вентиляторов для современных процессоров. При этом они должны будут отводить тепловую мощность всего несколько милливатт.

Для суперкомпьютерных центров можно создать более мощные нанокомпьютеры со стационарными криогенными установками.

С другой стороны, 3° К— это температура космоса. Почему бы космос с его неограниченными холодильными ресурсами не подключить к решению проблемы роста вычислительных ресурсов на Земле? Суперкомпьютерные центры, расположенные на геостационарных орбитах с дешевым космическим холодом, оснащенные мощными информационными каналами связи с Землей, — новое направление развития IT-бизнеса в будущем.

Хотя температуру рабочей среды компьютера можно еще понижать по сравнению с температурой жидкого гелия, при этом будут быстро расти и затраты на охлаждение.

Известны и другие физические механизмы, используя которые, можно оптимизировать термодинамику классического компьютера. Дело в том, что принцип Ландауэра выводится в предположении, что вычислительная среда характеризуется одной температурой *T*. Однако в физике известны среды с двумя и более температурами, т. е. являющиеся термодинамически неравновесными. Пример — всем хорошо известные газоразрядные лампы дневного света. Атомно-молекулярная подсистема здесь характеризуется комнатной температурой (300° К), а система свободных электронов —температурой в 30-50 раз большей (10000° К). Поэтому в вычислительной среде можно создавать переохлажденную рабочую подсистему с очень низкой температурой, а по завершении вычислительного процесса считывать результат еще до того, как начнут сказываться потери информации в результате возвращения системы к состоянию теплового равновесия. Последовательная реализация такого подхода приводит к идее оптимального сочетания возможностей квантового и классического компьютеров. Например, можно использовать взаимодействие холодных квантовых пучков легких частиц с массивами более теплых тяжелых частиц.

Таким образом, у СП-элементов имеется одновременное сочетание таких возможностей, как сверхвысокое быстродействие, низкое энергопотребление и идеальные характеристики соединений. Однако несмотря на все преимущества СП-элементов, реальной криогенной ЭВМ пока нет. Разработаны лишь отдельные микросхемы, узлы. Имеются отдельные проекты создания ЭВМ. Рассмотрим наиболее оригинальные из подобного рода разработок.

Фирмой IBM на джозефсоновских элементах спроектирован гипотетический процессор с архитектурой полупроводниковой ЭВМ IBM-3033. Если бы этот процессор был реализован, то производительность ЭВМ возросла бы с 4,9 до 70 млн оп/с, а суммарное потребление мощности и занимаемая площадь уменьшились бы соответственно с 190 кВт и 70 $м^2$ до 15 кВт и 16 $м^2$. Этой же фирмой в 1986 г. была предпринята попытка создать макет супер-ЭВМ со следующими характеристиками: производительность от 70 до 250 млн оп/с; емкость ОП до 16 Мбайт; размеры ЭВМ 1 5 х 1 5 х 5 см; потребляемая при температуре 4,2° К мощность 12,5 Вт. Для обеспечения этой мощности необходимо было иметь рефрижератор мощностью 30 КВт и стоимостью 30 тыс.долларов. Этот макет разрабатывали 115 исследователей, расходуя ежегодно до 20 млн долларов.

Однако поставленная задача не была решена, так как не удалось создать сверхбыстродействующее ЗУ емкостью 256 Кбайт для связи процессора с внешним ЗУ; ошибочно были выбраны материалы (легкоплавкие полупроводники), из-за чего не удалось создать надежные, долговечные, выдерживающие многократное термоциклирование туннельные джозефсоновские контакты; не была решена проблема воспроизводимости параметров элементов; не были отработаны вопросы по сборке и отладке ЭВМ.

Целый ряд фирм Японии (Hitachi, NEC, Fujitsu), а также ряд институтов и университетов занимались разработкой криогенной ЭВМ. В 1981 г. в Японии была принята программа по созданию в течение 8 лет ЭВМ с производительностью 100 MFLOPS в однопроцессорном варианте и 10 GFLOPS при максимальной конфигурации. При создании ЭВМ были использованы две альтернативные технологии: на охлаждаемых до -200° С элементах из арсенида галлия и на джозефсоновских элементах. В рамках этой программы японцам удалось решить проблему стабильности и устойчивости к многократному термоциклированию (более 1082 термоциклов, 4,2° К, -300° С), повысить качество, однородность и воспроизводимость характеристик, а также быстродействие джозефсоновских туннельных контактов (ТК) благодаря переходу от свинцовой технологии к ниобиевой.

Фирмой Fujitsu были разработаны логические элементы с задержкой 1,5 пс, и на их основе создана микросхема четырехразрядного микропроцессора. Микросхема имеет размер 5 х 5 мм и содержит 1841 логический вентиль. Подобные разработки были выполнены и другими фирмами. В целом можно сказать, что японскими фирмами были продемонстрированы рекордные характеристики по быстродействию и потребляемой мощности, но создать высокопроизводительную криогенную ЭВМ пока не удалось.

В связи с открытием высокотемпературной проводимости (ВСТП) было высказано немало прогнозов о том, что теперь уже не будет препятствий для создания криогенной супер-ЭВМ, поскольку жидкий азот доступнее, чем жидкий гелий. На ВСТП возлагаются большие надежды. Об этом говорят те ассигнования, которые выделяются на НИР в этой области. В США за 1988-1990 гг. они составили 548 млн долларов, в Японии в 1989 г. было выделено 88,9 млн долларов. По американским прогнозам, мировой рынок ВСТП-продукции к 2010 г. составит более 2 млрд долларов,

а по японским прогнозам он возрастет до 10 млрд долларов. Однако при практической реализации идей по использованию ВСТП нужны новые подходы для построения цифровых элементов. Поэтому появления криогенных ЭВМ на рынке СВТ следует ожидать не ранее чем через 10 лет.

3.5 Молекулярные компьютеры (клеточные и ДНК-процессоры)

В настоящее время в поисках реальной альтернативы полупроводниковым технологиям создания новых вычислительных систем ученые обращают все большее внимание на биотехнологии, или *биокомпьютинг,* который представляет собой гибрид информационных и молекулярных технологий. Биокомпьютинг позволяет решать сложные вычислительные задачи, пользуясь методами, принятыми в биохимии и молекулярной биологии, организуя вычисления с помощью живых тканей, клеток, вирусов и *биомолекул*. Наибольшее распространение получил подход, где в качестве основного элемента (процессора) используются молекулы дезоксирибонуклеиновой кислоты. Центральное место в этом подходе занимает так называемый *ДНК-процессор*. Кроме ДНК в качестве биопроцессора могут бы использованы также белковые молекулы и биологические мембраны.

ДНК-процессоры. как и любой другой процессор, характеризуется структурой и набором команд. В данном случае структура процессора — это структура молекулы ДНК, а набор команд — это перечень биохимических операций над молекулами. Принцип устройства компьютерной ДНК-памяти основан на последовательном соединении четырех нуклеотидов (основных кирпичиков ДНК-цепи). Три нуклеотида, соединяясь в любой последовательности, образуют элементарную ячейку памяти — *кодом*. Кодоны затем формируют цепь ДНК. Основная трудность в разработке ДНК-компьютеров связана с проведением избирательных *однокодонных* реакций (взаимодействий) внутри цепи ДНК. Однако прогресс есть уже и в этом направлении. Уже существует экспериментальное оборудование, позволяющее работать с одним из 1020 кодонов или молекул ДНК. Другой проблемой является *самосборка* ДНК, приводящая к потере информации. Ее преодолевают введением в клетку специальных ингибиторов — веществ, предотвращающих химическую реакцию самосборки.

Теоретическое обоснование подобной возможности было сделано еще в 50-х годах прошлого века (Р. П. Фейманом). В деталях теория была проработана в 70-х годах Ч. Бенеттом и в 80-х М. Конрадом. Первый компьютер на базе ДНК был создан в 1994 г. американским ученым Л. Адлеманом. Он смешал в пробирке молекулу ДНК, в которой были закодированы исходные данные и специальным образом подобранные ферменты. В результате химической реакции структура ДНК изменилась таким образом, что в ней в закодированном виде был представлен ответ задачи. Поскольку вычисления проводились в ходе химической реакции с участием ферментов, на них было затрачено очень мало времени.

Вслед за работой Адлемана последовали другие. Л. Смит из Университета Висконсин решил с помощью ДНК задачу доставки четырех сортов пиццы по четырем адресам, которая подразумевала 16 вариантов ответа. Ученые из Принстонского университета решили комбинаторную шахматную задачу: с помощью РНК нашли правильный ход шахматного коня на доске из девяти клеток (всего их 512 вариантов).

Р. Липтон из Принстона первым показал, как, используя ДНК, кодировать двоичные числа и решать проблему вычисления логического выражения. Суть ее в том, что, имея некоторое логическое выражение, включающее n логических переменных, нужно найти все комбинации значений переменных, делающих выражение истинным. Задачу можно решить только перебором 2^n комбинаций. Все эти комбинации легко закодировать с помощью ДНК, а дальше действовать по методике Адлемана. Липтон предложил также способ взлома шифра DES (американский криптографический) трактуемого как своеобразное логическое выражение.

Первую модель биокомпьютера, правда, в виде механизма пластмассы, в 1999 г. создал И. Шапиро из Вейцмановского института естественных наук. Она имитировала работу «молекулярной машины» в живой клетке, собирающей белковые молекулы по информации с ДНК, используя РНК в качестве посредника между ДНК и белком.

В 2001 г. Шапиро удалось реализовать вычислительное устройство на основе ДНК, которое может работать почти без вмешательства человека. Система имитирует машину Тьюринга -фундаментальных абстракций вычислительной техники, которая теоретически может решить любую вычислительную задачу. По своей природе

молекулы ДНК работают аналогичным образом, распадаясь и рекомбинируя в соответствии с информацией, закодированной в цепочках химических соединений. Разработанная установка кодирует входные данные и программы в состоящих из двух цепей молекулах ДНК и смешивает их с двумя ферментами.

Молекулы фермента выполняли роль аппаратного, а молекулы ДНК — программного обеспечения. Один фермент расщепляет молекулу ДНК с входными данными на отрезки разной длины в зависимости от содержащегося в ней кода, а другой — рекомбинирует эти отрезки в соответствии с их кодом и кодом молекулы ДНК с программой. Процесс продолжается вдоль входной цепи, и, когда доходит до конца, получается выходная молекула, соответствующая конечному состоянию системы.

Этот механизм может использоваться для решения самых разных задач. Хотя на уровне отдельных молекул обработка ДНК происходит медленно — с типичной скоростью от 500 до 1000 бит/с, что во много миллионов раз медленнее современных кремниевых процессоров, по своей природе она допускает массовый параллелизм. По оценкам Шапиро и его коллег, в одной пробирке может одновременно происходить триллион процессов, так что при потребляемой мощности в единицы нВт (10^{-9} ватт) может выполняться миллиард операций в секунду.

В конце февраля 2002 г. появилось сообщение, что фирма Olympus Optical претендует на первенство в создании коммерческо версии ДНК-компьютера, предназначенного для генетического ана лиза. Машина была создана в сотрудничестве с доцентом Токииск го университета А. Тояма.

Компьютер, построенный Olympus Optical, имеет молекулярную и электронную составляющие. Первая из них осуществляет химические реакции между молекулами ДНК, обеспечивает поиск и выделение результата вычислений, вторая — обрабатывает информацию и анализирует полученные результаты.

Клеточные компьютеры представляют собой самоорганизующиеся колонии различных «умных» микроорганизмов, в геном которых удалось включить некую логическую схему, которая могла бы активизироваться в присутствии определенного вещества. Для этой цели идеально подошли бы бактерии, стакан с которыми и представлял бы собой компьютер. Главным свойством компьютера такого рода является то, что каждая их клетка представляет собой миниатюрную химическую лабораторию. Если биоорганизм запрограммирован, то он просто производит

нужные вещества. Достаточно вырастить одну клетку, обладающую заданными качествами, и можно легко и быстро вырастить тысячи клеток с такой же программой.

Основная проблема, с которой сталкиваются создатели клеточных биокомпьютеров, — организация всех клеток в единую работающую систему. На сегодняшний день практические достижения в области клеточных компьютеров напоминают достижения 20-х гг. в области ламповых и полупроводниковых компьютеров. Сейчас в Лаборатории искусственного интеллекта Массачусетского технологического университета создана клетка, способная хранить на генетическом уровне 1 бит информации. Также разрабатываются технологии, позволяющие единичной бактерии отыскивать своих соседей, образовывать с ними упорядоченную структуру и осуществлять массив параллельных операций.

В 2001 г. американские ученые создали трансгенные микроорганизмы (т. е. микроорганизмы с искусственно измененными генами), клетки которых могут выполнять логические операции *И* и *ИЛИ*.

Специалисты лаборатории Оук-Ридж, штат Теннесси, использовали способность генов синтезировать тот или иной белок под воздействием определенной группы химических раздражителей. Ученые изменили генетический код бактерий *Pseudomonas putida* таким образом, что их клетки обрели способность выполнять простые логические операции. Например, при выполнении операции *И* в клетку подаются два вещества (по сути — входные операнды), под влиянием которых ген вырабатывает определенный белок. Теперь ученые пытаются создать на базе этих клеток более сложные логические элементы, а также подумывают о возможности создания клетки, выполняющей параллельно несколько логических операций.

Потенциал биокомпьютеров очень велик. К достоинствам, вы годно отличающим их от компьютеров, основанных на кремниниевых технологиях, относятся:

- более простая технология изготовления, не требующая для своей реализации столь жестких условий, как при производстве полупроводников;
- использование не бинарного, а тернарного кода (информация кодируется тройками нуклеотидов), что позволит при меньшем количестве шагов перебрать большее число вариантов при анализе сложных систем;

- потенциально исключительно высокая производительность, которая может составлять до 10^{14} операций в секунду за счет одновременного вступления в реакцию триллионов молекул ДНК;
- возможность хранить данные с плотностью, во много раз превышающей показатели оптических дисков;
- исключительно низкое энергопотребление.

Однако, наряду с очевидными достоинствами, биокомпьютеры имеют и существенные недостатки, такие, как:

- сложность со считыванием результатов — современные способы определения кодирующей последовательности не совершенны, сложны, трудоемки и дороги;
- низкая точность вычислений, связанная с возникновением мутаций, прилипанием молекул к стенкам сосудов и т. д.;
- невозможность длительного хранения результатов вычислений в связи с распадом ДНК в течение времени.

Хотя до практического использования биокомпьютеров еще очень далеко, и они вряд ли будут рассчитаны на широкие массы пользователей, предполагается, что они найдут достойное применение в медицине и фармакологии, а также с их помощью станет возможным объединение информационных и биотехнологий. Вероятно, в будущем их смогут использовать не только для вычислений, но и как своеобразные «нанофабрики» лекарств. Поместив подобное «устройство» в клетку, врачи смогут влиять на ее состояние, исцеляя человека от самых опасных недугов.

3.6 Коммуникационные компьютеры

Коммуникационные компьютеры строятся на процессорах, представляющие собой микрочипы, являющие собой нечто среднее между жесткими специализированными интегральными микросхемами и гибкими процессорами общего назначения. Коммуникационные процессоры программируются, как и привычные ПК-процессоры, но построены с учетом сетевых задач, оптимизированы для сетевой работы, и на их основе производители (как процессоров, так и другого оборудования) создают программное обеспечение для специфических приложений. Коммуникационный процессор имеет собственную память и оснащен высокоскоростными внешними каналами для соединения с другими процессорными узлами. Его

присутствие позволяет в значительной мере освободить вычислительный процессор от нагрузки, связанной с передачей сообщений между процессорными узлами. Скоростной коммуникационный процессор с RISC-ядром позволяет управлять обменом данными по нескольким независимым каналам, поддерживать практически все распространенные протоколы обмена, гибко и эффективно распределять и обрабатывать последовательные потоки данных с временным разделением каналов.

Сама идея создания процессоров, предназначенных для оптимизации сетевой работы — и при этом достаточно универсальных для программной модификации, — родилась в связи с необходимостью устранить различия в подходах к созданию локальных сетей (различные подходы к архитектуре сети, классификации потоков и т. д.). Несомненно, истинной причиной бума сетевых процессоров стало ускорение темпов развития рынка. Когда рынок движется на «Internet-скорости», поставщики оборудования уже не могут тратить по два года на разработку специализированных микросхем для реализации конкретных сетевых функций. Эти два года (и вложенные деньги) будут потрачены зря, если рынок за это время уйдет в другом направлении. Выход один — разрабатывать процессоры, которые поставщики оборудования могут внедрить и выпустить в новом ' продукте в течение нескольких месяцев. Бум сетевых процессоров, окончательно оформившийся в середине 1999 г., не был кратким, и в последующие годы индустрия развивалась крайне бурно.

По прогнозам, рынок коммуникационных процессоров в 2004 г. составит около 2,9 млрд долл. и с увеличением объемов специальные микросхемы будут вытеснены стандартными сетевыми процессорами. Более «умеренные» аналитики считают, что у сетевых процессоров, без сомнения, есть будущее, но они смогут преобладать только на некоторых сегментах рынка, где необходимы укороченные циклы разработки, быстрота и гибкость.

Прогнозируется, что на этом рынке не будет преобладать какая-либо одна компания, как, например, Intel на рынке ПК. Однако считается, что Intel все же будет одним из ключевых игроков, разделив 2,9 млрд долл. с IBM, Motorola и дюжиной других компаний.

Серия коммуникационных процессоров INTEL IXP4xx построена на базе распределенной архитектуры XScale и включает мультимедийные возможности, а также развитые сетевые интерфейсы Ethernet. Сочетание высокой

производительности и низкого энергопотребления позволяет эффективно применять коммуникационные процессоры INTEL не только в классических сетевых приложениях, но и для построения Internet-ориентированных встраиваемых систем промышленного назначения.

Эффективность работы промышленных предприятий сегодня напрямую зависит от гибкости применяемых систем автоматизированного управления. Крупные производственные установки требуют использования нескольких децентрализованных систем управления связанных друг с другом мощной информационной сетью, способной работать в сложных промышленных условиях. Зачастую эти средства промышленной коммуникации призваны обеспечить возможность гибкого управления, программирования и контроля работы распределенных систем управления из удаленных диспетчерских пунктов. Осуществление этих целей возможно с помощью коммуникационных процессоров, предназначенных для подключения персональных компьютеров к промышленным информационным сетям.

Дополнительные возможности, обеспечиваемые коммуникационными процессорами, должны быть интересны, прежде всего, тем пользователям, которым необходимо осуществлять сложные транзакции или наладить прямую голосовую и видеопередачу в рамках сетевой инфраструктуры.

3.7 Компьютеры баз данных

Компьютерами (процессорами) *баз данных* принято называть программно-аппаратные комплексы, предназначенные для выполнения всех или некоторых функций систем управления базами данных (СУБД). Если в свое время системы управления базами данных предназначались в основном для хранения текстовой и числовой информации, то теперь они рассчитаны на самые различные виды данных, в том числе графические, звуковые и видео. Процессоры баз данных выполняют функции управления и распространения, обеспечивают дистанционный доступ к информации через шлюзы, а также репликацию обновленных данных с помощью различны механизмов тиражирования.

Современные процессоры баз данных должны обеспечивать естественную связь накапливаемой в базах данных информации средствами оперативной обработки транзакций и Internet-приложениями. Это должны быть системы, которые дают

пользователям возможность в любой момент обратиться к корпоративным данным и проанализировать их вне зависимости от того, где эти данные размещаются.

Решение таких задач требует существенного увеличения производительности таких систем. Однако традиционная программная реализация многочисленных функций современных СУБД на ЭВМ общего назначения приводит к громоздким и непроизводительным системам с недостаточно высокой надежностью. Необходим поиск новых архитектурных и аппаратных решений. Интенсивные исследования, проводимые в этой области, привели к пониманию необходимости использования в качестве процессоров баз данных специализированных параллельных вычислительных систем. Создание такого рода систем связывается с реализацией параллелизма при выполнении последовательности операций и транзакций, а также конвейерной потоковой обработки данных.

3.8 Нейронные компьютеры

Одно из наиболее перспективных направлений разработки принципиально новых архитектур вычислительных систем тесно связано с созданием компьютеров нового поколения на основе принципов обработки информации, заложенных в искусственных нейронных сетях (НС).

Первые практические работы по искусственным нейросетям и нейрокомпьютерам начались еще в 40—50-е гг. Под *нейронной сетью* обычно понимают совокупность элементарных преобразователей информации, называемых *нейронами,* которые определенным образом соединены друг с другом каналами обмена информации *синоптическими связями.*

Одним из основных достоинств нейровычислителя является то, что его основу составляют относительно простые, чаще всего однотипные элементы, имитирующие работу нейронов мозга. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены или заторможены. Он обладает группой *синапсов* — однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет *аксон* — выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Общий вид нейрона приведен на рис. 3.7, а.

Каждый синапс характеризуется *величиной синоптической связи* или ее весом w,, который по физическому смыслу эквивалентен электрической проводимости. Текущее состояние нейрона определяется как взвешенная сумма его входных сигналов:

$$S = \sum_{i=1}^{n} x_i w_i$$

Выход нейрона есть функция его состояния: `y=f(s)`, которая называется *активационной*. Известны различные виды таких функций, некоторые из которых представлены на рис. 3.8. Одной из наиболее распространенных является нелинейная функция с насыщением, так называемая сигмоидальная (логистическая) функция:

$$f(x) = \frac{1}{1 + e^{-ax}}$$

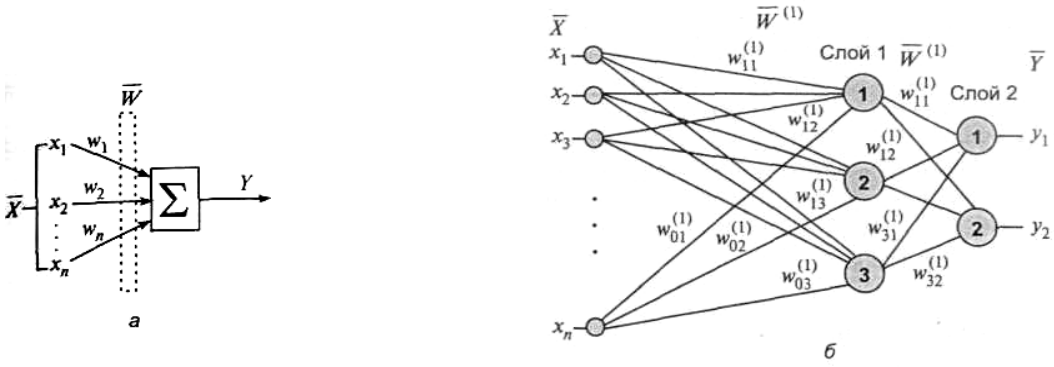


Рисунок 3.7 - Нейрон *(а)* и нейросеть *(б)*

Состояния нейронов изменяются в процессе функционирования и составляют кратковременную память нейросети. Каждый нейрон вычисляет взвешенную сумму пришедших к нему по синапсам сигналов и производит над ней нелинейное преобразование. При пересылке по синапсам сигналы умножаются на некоторый ве совой коэффициент. В распределении весовых коэффициентов за ключается информация, хранимая в ассоциативной памяти НС При обучении и переобучении НС ее весовые коэффициенты изменяются. Однако они остаются постоянными при функционировании нейросети, формируя долговременную память.

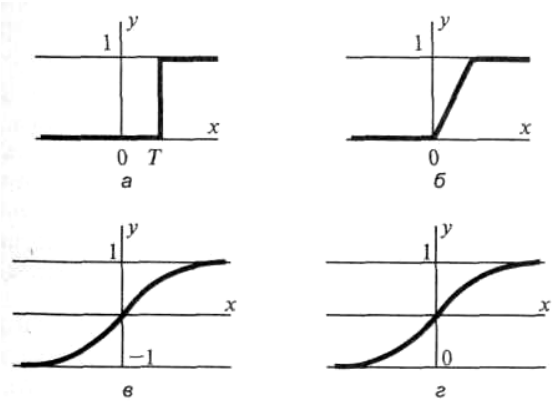


Рисунок 3.8 - Типовые активационные функции: а) — единичная пороговая функция; *б* — линейный порог (гистерезис); *в* — сигмоид (гиперболический тангенс); *г* — логистический сигмоид

НС может состоять из одного слоя, из двух и большего числа однако, как правило, для решения практических задач более трех слоев в НС не требуется. Число входов НС определяет размерность гиперпространства, в котором входные сигналы могут быть представлены точками или гиперобластями из близко расположенных точек. Количество нейронов в слое сети определяет число гиперплоскостей в гиперпространстве. Вычисление взвешенных сумм и выполнение нелинейного преобразования позволяют определить, с какой стороны от той или иной гиперплоскости в гиперпространстве находится точка входного сигнала.

В нейрокомпьютерах используются принципы обработки информации, осуществляемые в реальных нейронных сетях. Это принципиально новые вычислительные средства с нетрадиционной архитектурой позволяют выполнять высокопроизводительную обработку информационных массивов большой размерности. В отличие от традиционных вычислительных систем нейросетевые вычислители, аналогично нейронным сетям, дают возможность с большей скоростью обрабатывать информационные потоки дискретных и непрерывных сигналов, содержат простые вычислительные элементы и с высокой степенью надежности позволяют решать информационные задачи обработки данных, обеспечивая при этом режим самоперестройки вычислительной среды в зависимости от полученных решений.

Вообще говоря, под термином *нейрокомпьютер* подразумевается довольно широкий класс вычислителей. Это происходит по той причине, что формально нейрокомпьютером можно считать любую аппаратную реализацию нейросетевого алгоритма от простой модели биологического нейрона до системы распознавания

символов или движущихся целей. Нейрокомпьютеры не являются компьютерами в общепринятом смысле этого слова. В настоящее время технология еще не достигла того уровня развития, при котором можно было бы говорить о нейрокомпьютере общего назначения (который являлся бы одновременно искусственным интеллектом). Системы с фиксированными значениями весовых коэффициентов — вообще самые узкоспециализированные из нейросетевого семейства. Обу чающиеся сети более гибки к разнообразию решаемых задач. Таким образом, построение нейрокомпьютера — это каждый раз широчайшее поле для исследовательской деятельности в области аппаратной реализации практически всех элементов НС.

В то же время технология интегральной электроники близка к исчерпанию своих физических возможностей. Геометрические раз-меры транзисторов больше нельзя физически уменьшать: при тех-нотогически достижимых размерах порядка 1 мкм и меньше прояв-ляются физические явления, незаметные при больших размерах активных элементов — начинают сильно сказываться квантовые **размерные** эффекты. Транзисторы перестают работать как транзисторы.

Для аппаратной реализации НС необходим новый носитель информации. Таким новым носителем информации может быть свет, который позволит резко, на несколько порядков, повысить производительность вычислений.

Единственной технологией аппаратной реализации НС, способной в будущем прийти на смену оптике и оптоэлектронике, является нанотехнология, способная обеспечить не только физически предельно возможную степень интеграции субмолекулярных квантовых элементов с физически предельно возможным быстродействием, но и столь необходимую для аппаратной реализации НС трехмерную архитектуру.

Длительное время считалось, что нейрокомпьютеры эффективны для решения так называемых неформализуемых и плохо формализуемых задач, связанных с необходимостью включения в алгоритм решения задачи процесса обучения на реальном экспериментальном материале. В первую очередь к таким задачам относилась задача аппроксимации частного вида функций, принимающих дискретное множество значений, т. е. задача распознавания образов.

В настоящее время к этому классу задач добавляется класс задач, иногда не требующий обучения на экспериментальном материале, но хорошо представимый в

нейросетевом логическом базисе. К ним относятся задачи с ярко выраженным естественным параллелизмом обработки сигналов, обработка изображений и др. Подтверждением точки зрения, что в будущем нейрокомпьютеры будут более эффективными, чем прочие архитектуры, может, в частности, служить резкое расширение в последние годы класса обшематематических задач, решаемых в нейросетевом логическом базисе. К ним, кроме перечисленных выше, можно отнести задачи решения линейных и нелинейных алгебраических уравнений и неравенств большой размерности; систем нелинейных дифференциальных уравнений; уравнений в частных производных; задач оптимизации и других задач.

3.9 Компьютеры с многозначной (нечеткой) логикой

Идея построения процессоров с *нечеткой логикой (fuzzy logic)* основывается на *нечеткой математике.* Основанные на этой теории различные компьютерные системы, в свою очередь, существенно расширяют область применения нечеткой логики.

Подходы нечеткой математики дают возможность оперировать входными данными, непрерывно меняющимися во времени, и значениями, которые невозможно задать однозначно, такими, например, как результаты статистических опросов. В отличие от традиционной формальной логики, известной со времен Аристотеля и оперирующей точными и четкими понятиями типа «истина» и «ложь» «да» и «нет», «0» и «1», нечеткая логика имеет дело со значениями, лежащими в некотором (непрерывном или дискретном) диапазоне (рис. 3.9).

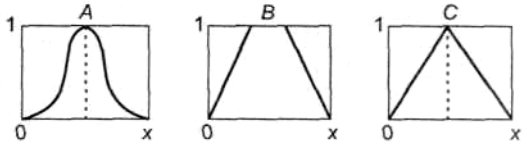


Рисунок 3.9 - Различные типы функций принадлежности

Функция принадлежности элементов к заданному множеству также представляет собой не жесткий порог «принадлежит — не принадлежит», а линию, проходящую все значения от нуля до единицы. Теория нечеткой логики позволяет выполнять над такими величинами все логические операции — объединение, пересечение, отрицание и др. (рис. 3.10).

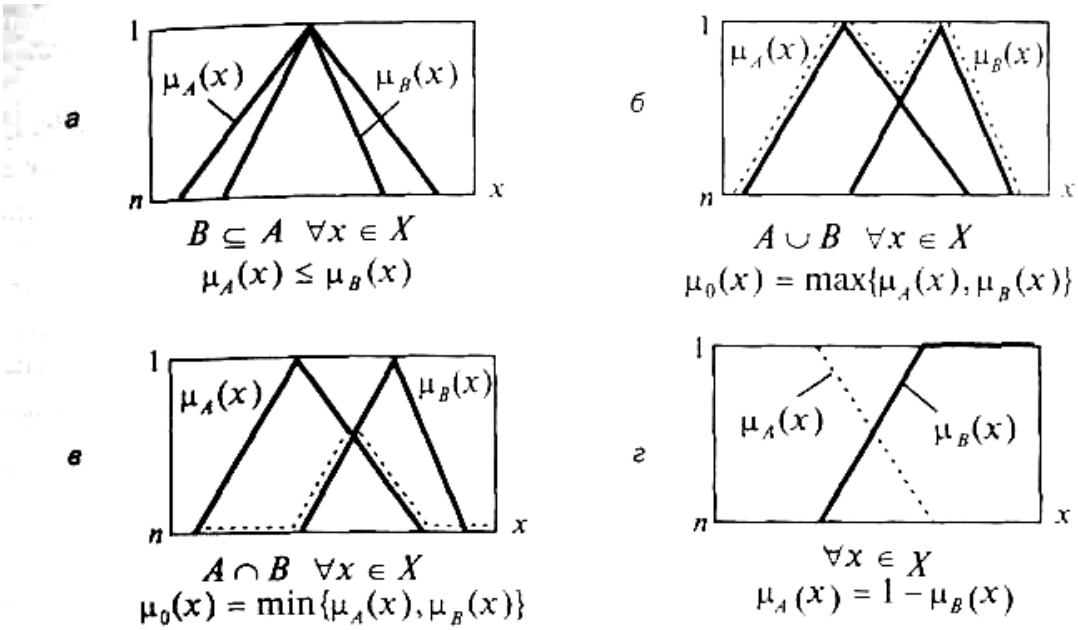


Рисунок 3.10 - Операции включения: *а)* объединения; б) пересечения; в) и дополнения; г) НМ

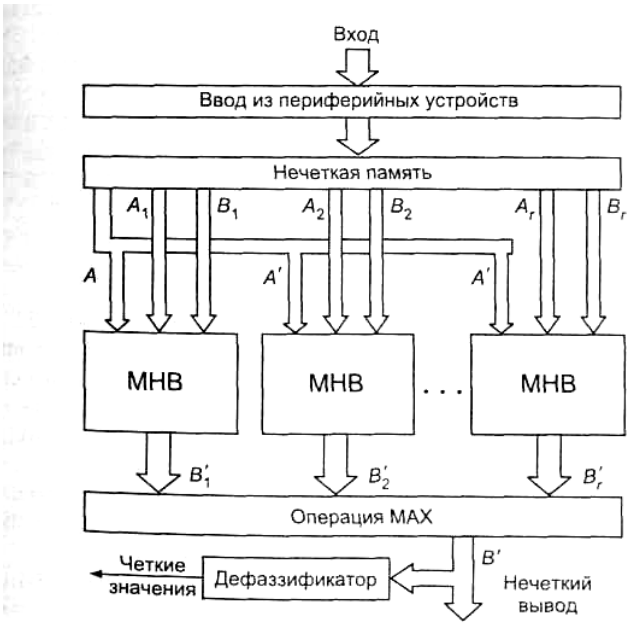


Рисунок 3.11 - Архитектура нечеткого компьютера: МНВ — механизм нечеткого вывода

Задачи с помощью нечеткой логики решаются по следующему принципу (рис. 3.11):

1) численные данные (показания измерительных приборов, результаты анкетирования) *фаззируются* (переводятся в нечеткий формат);

2) обрабатываются по определенным правилам;

3) *дефаззируются* и в виде привычной информации подаются на выход.

В 1986 г. в AT&T Bell Labs создавались процессоры с «прошитой» нечеткой логикой обработки информации. В начале 90-х компания Adaptive Logic из США выпустила кристалл, сделанный по аналого-цифровой технологии. Он позволит сократить сроки конструирования многих встроенных систем управления реального вре мени, заменив собой традиционные схемы нечетких микроконтроллеров. Аппаратный процессор нечеткой логики второго поколения принимает аналоговые сигналы, переводит их в нечеткий формат затем, применяя соответствующие правила, преобразует результаты в формат обычной логики и далее — в аналоговый сигнал. Все это осуществляется без внешних запоминающих устройств, преобразователей и какого бы ни было программного обеспечения нечеткой логики.

В Европе и США ведутся интенсивные работы по интеграции *fuzzy-команд,* в ассемблеры промышленных контроллеров встроенных устройств (чипы Motorola 68HC11. 12.21). Такие аппаратные средства позволяют в несколько раз увеличить скорость выполнения приложений и компактность кода по сравнению с реализацией на обычном ядре. Кроме того, разрабатываются различные варианты fuzzy-сопроцессоров, которые контактируют с центральным процессором через общую шину данных, концентрируют свои усилия на размывании/уплотнении информации и оптимизации использования правил (продукты Siemens Nixdorf).

Нечеткая логика не решит всех тех задач, которые не решаются на основе логики двоичной, но во многих случаях она удобнее, производительнее и дешевле. Разработанные на ее основе специализированные аппаратные решения (fuzzy-вычислители) позволят получить реальные преимущества в быстродействии. Если каскадировать fuzzy-вычислители, получается один из вариантов нейропроцессора или нейронной сети. Во многих случаях эти понятия объединяют, называя общим термином *«neuro-fuzzy logic».*

Вопросы для самопроверки

1. Охарактеризуйте одиночный поток команд — одиночный поток данных (ОКОД).
2. Охарактеризуйте одиночный поток команд — множественный поток данных (ОКМД).
3. Охарактеризуйте множественный поток команд — одиночный поток данных (МКОД).
4. Охарактеризуйте множественный поток команд — множественный поток данных (МКМД).
5. Чем многомашинные ВС отличаются от многопроцессорных?
6. Приведите характеристику каждого из четырех классов архитектуры ВС согласно классификации по режиму выполнения?
7. Какие уровни комплексирования ЭВМ вам известны?
8. Чем отличаются многомашинные ВС от многопроцессорных ВС?
9. На какие классы подразделяются многопроцессорные параллельные ВС?
10. Что такое кластеры и какими преимуществами они обладают?
11. Что такое коммутационные среды? Приведите примеры коммутаторов.
12. Охарактеризуйте стратегии управления иерархической памятью.
13. Что такое когерентность памяти? И когда она возникает?
14. Что такое вычислительные системы и каковы их разновидности?
15. Охарактеризуйте принципы функционирования машин типа wavefront и reduction.
16. Назовите основные классы и подклассы вычислительных машин и дайте их сравнительную характеристику.
17. Дайте общую характеристику и определите область использования суперЭВМ и мэйнфреймов.
18. Дайте общую характеристику потоковых ВС.
19. Дайте общую характеристику ассоциативных ВС.
20. Назовите перспективные направления развития вычислительной техники?
21. В чем разница между нанокомпьютером и нейрокомпьютером?
22. Назовите основные особенности компьютеров с нечеткой логикой?
23. Матричные и векторные ВС. Основные различия.
24. Что такое транспьютер?
25. Криогенные и оптические компьютеры.
26. Назовите системы класса SIMD?
27. Назовите системы класса MIMD?
28. Дайте общую характеристику симметричных ВС.
29. Какие архитектуры SMP вы знаете?
30. Как построены систолические ВС?
31. Назовите топологии кластеров?
32. Чем характеризуется VLIW-вычислительные системы?

Типовые варианты тестовых вопросов

Вопрос № 1

Какой из принципов фон-неймановской концепции вычислительной машины можно рассматривать в качестве наиболее существенного?

Варианты ответов

1. Двоичного кодирования
2. Микропрограмного управления
3. Однородности памяти
4. Адресности
5. Дискретности

Вопрос № 2

На каком принципе основаны матричные ВС?

Варианты ответов

1. SIMD 2. MIND 3. RISC 4. CISC 5. DRAM

Вопрос № 3

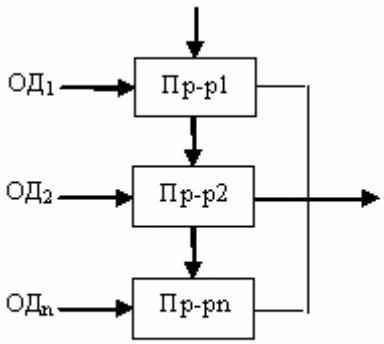
Что есть совокупность характеристик и параметров, определяющих функциональную и структурную организацию системы, структуру обрабатываемых данных?

Варианты ответов

1. Система 2. Качество 3. Производительность
4. Быстродействие 5. Архитектура

Вопрос №4

Укажите архитектуру ВС?



Варианты ответов

1. ВКВД 2. ОКОД 3. МКОД 4. ОКМД 5. МКМД

Вопрос №5

Сколько существует уровней комплексирования машин в вычислительную систему?

Варианты ответов

1. Два 2. Десять 3. Восемь 4. Три 5. Пять

Вопрос №6

В какой памяти возникает проблема когерентности?

Варианты ответов

1. В одноуровневой иерархической памяти
2. В многоуровневой иерархической памяти
3. В виртуальной памяти
4. В стековой памяти
5. В регистрах

Вопрос №7

По принципу закрепления вычислительных функций за отдельными ЭВМ (процессорами) различают ВС с…?

Варианты ответов

1. Асинхронным
2. Жестким
3. Плавающим
4. Временным
5. Синхронным

Вопрос №8

Как разделяются вычислительные системы по типу ЭВМ или процессоров?

Варианты ответов

1. Многомашинные
2. Однородные
3. Многопроцессорные
4. Неоднородные
5. Централизованные

Вопрос №9

Как называются вычислительные системы, использующие общую разделяемую память для межпроцессорного взаимодействия и синхронизации?

Варианты ответов

1. Архитектурами с разделяемой памятью
2. Архитектурами с общей памятью
3. Архитектурами с передачей сообщений
4. Гибридными архитектурами
5. Архитектурами с общим процессором

Вопрос №10

В какой памяти возникает проблема когерентности?

Варианты ответов

1. В одноуровневой иерархической памяти
2. В многоуровневой иерархической памяти
3. В виртуальной памяти
4. В стековой памяти
5. В регистрах

Список рекомендуемой литературы

1. Дрейвс Ю.Г. Организация ЭВМ и вычислительных систем: Учебник для вузов/Ю.Г. Дрейвс. – М.: Высшая школа, 2006. – 501 с.

2. Вычислительные системы, сети и телекоммуникации: Учебник. - 2-ое изд., перераб. и доп./ А.П. Пятибратов, Л.П. Гудыно, А.А. Кириченко; Под ред. А.П. Пятибратова. – М.: Финансы и статистика, 2004. – 512 с.

3. Нанотехнологии. Азбука для всех./ Под.ред. Ю.Д.Третьякова. 2-изд. перераб. – М.:Физматлит, 2009. – 368 с.

4. Рамбиди Н.Г. Нанотехнологии и молекулярные компьютеры. - М.:Физматлит, 2007. – 256 с.

5. Горнец Н.Н. Организация ЭВМ и систем: учеб. пособие для студ. высш. учеб. заведений/ Н.Н. Горнец, А.Г. Рощин, В.В. Соломенцев. – 2-е изд., стер. – М.: Издательский центр «Академия», 2008. – 320 с.

6. Тихонов В.А., Баранов А.В. Организация ЭВМ и систем: Учебник/ Под ред. акад. В.К. Левина. – М.: Гелиос АРВ, 2008. – 400 с.

7. Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: Учебник для ВУЗов. – СПБ.: Питер, 2004. – 668 с.

8. Попов И. И., Максимов Н. В., Партыка Т. Л. Архитектура ЭВМ вычислительных систем. Учебник. – М.: Инфра-М, Форум, 2010. - 512 с.

Учебное издание

Михайлов Борис Михайлович
Халабия Рустам Фарук

КЛАССИФИКАЦИЯ И ОРГАНИЗАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Учебное пособие

ЛР № 020418 от 08 октября 1997 г.
Подписано в печать _____
Формат 60x84 1/16
Объем 3,2 п.л. Тираж 100 экз. Заказ № _____

ГОУ ВПО "Московский государственный университет приборостроения и информатики"
107996, Москва, ул. Стромынка, 20